Semi-Oblivious Chase Termination: The Sticky Case

Theory of Computing Systems (2021) 65:84–121
https://doi.org/10.1007/s00224-020-09994-5

Semi-Oblivious Chase Termination: The Sticky Case

Marco Calautti[1] · **Andreas Pieris**[2]

Published online: 17 August 2020

Abstract
The chase procedure is a fundamental algorithmic tool in database theory with a variety of applications. A key problem concerning the chase procedure is all-instances termination: for a given set of tuple-generating dependencies (TGDs), is it the case that the chase terminates for every input database? In view of the fact that this problem is undecidable, it is natural to ask whether known well-behaved classes of TGDs, introduced in different contexts such as ontological reasoning, ensure decidability. We consider a prominent paradigm that led to a robust TGD-based formalism, called stickiness. We show that for sticky sets of TGDs, all-instances chase termination is decidable if we focus on the (semi-)oblivious chase, and we pinpoint its exact complexity: PSPACE-complete in general, and NLOGSPACE-complete for predicates of bounded arity. These complexity results are obtained via a graph-based syntactic characterization of chase termination that is of independent interest.

Keywords Chase procedure · Tuple-generating dependencies · Stickiness · Termination · Computational complexity

1 Introduction

The *chase procedure* (or simply chase) is a fundamental algorithmic tool that has been successfully applied to several database problems such as containment of queries under constraints [2], checking logical implication of constraints [5, 27],

This article belongs to the Topical Collection: *Special Issue on Database Theory (ICDT 2019)*
Guest Editor: Pablo Barceló

✉ Andreas Pieris
apieris@inf.ed.ac.uk

Marco Calautti
marco.calautti@unitn.it

[1] Department of Information Engineering and Computer Science, University of Trento, Trento, Italy

[2] School of Informatics, University of Edinburgh, Edinburgh, UK

computing data exchange solutions [17], and query answering under constraints [11], to name a few. The chase procedure takes as input a database D and a set Σ of constraints, which, for this work, are *tuple-generating dependencies (TGDs)* of the form

$$\forall \bar{x} \forall \bar{y} (\phi(\bar{x}, \bar{y}) \rightarrow \exists \bar{z} \psi(\bar{x}, \bar{z})),$$

where ϕ (the body) and ψ (the head) are conjunctions of relational atoms, and it produces an instance D_Σ that is a *universal model* of D and Σ, i.e., a model that can be homomorphically embedded into every other model of D and Σ. Somehow D_Σ acts as a representative of all the models of D and Σ. This is the reason for the ubiquity of the chase in database theory, as discussed in [15]. Indeed, many database problems can be solved by simply exhibiting a universal model.

1.1 The Chase in a Nutshell

Roughly, the chase adds new tuples to the database D (possibly involving null values that act as witnesses for the existentially quantified variables), as dictated by the TGDs of Σ, and it keeps doing this until all the TGDs of Σ are satisfied. There are, in principle, three different ways for formalizing this simple idea, which lead to different versions of the chase procedure:

Oblivious Chase The first one, which gives rise to the *oblivious chase*, is as follows: for each pair $(\bar{t}, \bar{u})$ of tuples of terms from the instance I constructed so far, apply a TGD σ of the form $\forall \bar{x} \forall \bar{y} (\phi(\bar{x}, \bar{y}) \rightarrow \exists \bar{z} \psi(\bar{x}, \bar{z}))$ if $\phi(\bar{t}, \bar{u}) \subseteq I$, and σ has not been applied in a previous chase step due to the same pair $(\bar{t}, \bar{u})$, and add to I the set of atoms $\psi(\bar{t}, \bar{v})$, where $\bar{v}$ is a tuple of new nulls not occurring in I.

Semi-oblivious Chase The second one, which is a refinement of the oblivious chase, and it gives rise to the *semi-oblivious chase*, is as follows: for each pair $(\bar{t}, \bar{u})$ of tuples of terms from the instance I constructed so far, apply a TGD σ of the form $\forall \bar{x} \forall \bar{y} (\phi(\bar{x}, \bar{y}) \rightarrow \exists \bar{z} \psi(\bar{x}, \bar{z}))$ if $\phi(\bar{t}, \bar{u}) \subseteq I$, and σ has not been applied in a previous chase step due to a pair of tuples $(\bar{t}, \bar{u}')$ with $\phi(\bar{t}, \bar{u}') \subseteq I$, where $\bar{u}$ and $\bar{u}'$ might be different, and add to I the set of atoms $\psi(\bar{t}, \bar{v})$, where $\bar{v}$ is a tuple of new nulls not in I. In other words, a TGD σ of the above form is applied only once due to a certain witness $\bar{t}$ for the variables $\bar{x}$ that appear both in ϕ and ψ.

Restricted Chase The third one, which is a refinement of the (semi-)oblivious chase that leads to the *restricted* (a.k.a. *standard*) *chase*, is as follows: for each pair $(\bar{t}, \bar{u})$ of tuples of terms from the instance I constructed so far, apply a TGD σ of the form $\forall \bar{x} \forall \bar{y} (\phi(\bar{x}, \bar{y}) \rightarrow \exists \bar{z} \psi(\bar{x}, \bar{z}))$ if $\phi(\bar{t}, \bar{u}) \subseteq I$, and there is no tuple $\bar{u}'$ of terms from I such that $\psi(\bar{t}, \bar{u}') \subseteq I$, i.e., the TGD is not already satisfied, and add to I the set of atoms $\psi(\bar{t}, \bar{v})$, where $\bar{v}$ is a tuple of new nulls not in I.

Thus, the key difference between the (semi-)oblivious and restricted versions of the chase is that the former apply a TGD whenever the body is satisfied, while the latter applies a TGD if the body is satisfied but the head is not.

1.2 Restricted vs. (Semi-)Oblivious Chase

It should not be difficult to verify that the restricted chase, in general, builds smaller instances than the (semi-)oblivious one. In fact, it is easy to devise an example where, according to the restricted chase, none of the TGDs should be applied, while the (semi-)oblivious chase builds an infinite instance.

Example 1 Consider the database $D = \{R(a, a)\}$ and the TGD

$$\forall x \forall y (R(x, y) \rightarrow \exists z R(z, x)).$$

The restricted chase will detect that the database already satisfies the TGD, while the (semi-)oblivious chase will build the infinite instance

$$\{R(a, a), R(\bot_1, a), R(\bot_2, \bot_1), R(\bot_3, \bot_2), \ldots\},$$

where $\bot_1, \bot_2, \bot_3, \ldots$ are (labeled) nulls.

It should be also clear that the semi-oblivious chase, in general, builds smaller instances than the oblivious one. This is illustrated by the following example.

Example 2 Consider the database $D = \{R(a, a)\}$ and the TGD

$$\forall x \forall y (R(x, y) \rightarrow \exists z R(x, z)),$$

The semi-oblivious chase will build the instance $\{R(a, a), R(a, \bot)\}$, where $\bot$ is a null, whereas the oblivious chase will build the infinite instance

$$\{R(a, a), R(a, \bot_1), R(a, \bot_2), R(a, \bot_3), \ldots\},$$

where $\bot_1, \bot_2, \bot_3, \ldots$ are nulls.

The restricted chase has a clear advantage over the (semi-)oblivious chase when it comes to the size of the final result. But, of course, this advantage does not come for free: at each application, the restricted chase has to check that there is no way to satisfy the head of the TGD at hand, and this can be computationally costly in practice. On the other hand, the advantage of the semi-oblivious chase over the oblivious chase comes without any additional overhead since both versions have to keep track of the TGDs and pairs of tuples that have been considered so far.

It has been recently observed that for RAM-based implementations the restricted chase is the indicated approach since the benefit from producing smaller instances justifies the additional effort for checking whether a TGD is already satisfied; see, e.g., [6, 22]. However, as discussed in [6], an RDBMS-based implementation of the restricted chase is quite challenging, whereas an efficient implementation of the semi-oblivious chase is feasible. Hence, both the semi-oblivious and restricted versions of the chase are relevant algorithmic tools for practical implementations, whereas the oblivious version of the chase is mostly of theoretical interest.

1.3 The Challenge of Non-Termination

There are indeed efficient implementations of the semi-oblivious and restricted chase that allow us to solve central database problems by adopting a materialization-based approach [6, 22, 30, 33]. Nevertheless, for this to be feasible in practice we need a guarantee that the chase terminates, which is not always the case. This fact motivated a long line of research on identifying classes of TGDs that ensure the termination of the semi-oblivious and/or restricted chase, no matter how the input database looks like. A prime example is the class of *weakly-acyclic* sets of TGDs [17], which has been introduced in the context of data exchange, and guarantees the termination of both the semi-oblivious and restricted chase. Many other termination criteria can be found in the literature; see, e.g., [4, 9, 14–16, 21, 23, 28, 29].

With so much effort spent on identifying sufficient conditions for the termination of the chase, the question that immediately comes up is whether a sufficient condition that is also *necessary* exists. In other words, given a set Σ of TGDs, is it possible to decide whether, for every database D, the semi-oblivious or the restricted chase on D and Σ terminates? The answer is negative, no matter which version of the chase we consider [18]; this is actually true even for the oblivious version of the chase. The problem remains undecidable even if the database is known; this has been established in [15] for the restricted chase, and it was observed in [28] that the same proof shows undecidability also for the (semi-)oblivious chase.

1.4 Deciding the Termination of the Chase

The undecidability proof given in [18] constructs a sophisticated set of TGDs that goes beyond existing well-behaved classes of TGDs that enjoy certain syntactic properties, which in turn ensure useful model-theoretic properties. Such well-behaved classes of TGDs have been proposed in the context of ontological reasoning. The two main paradigms that led to robust TGD-based formalisms, without forcing the chase to terminate, are *guardedness* [4, 11, 12] and *stickiness* [13]:

Guardedness A TGD is guarded if its body has an atom that contains (or "guards") all the universally quantified variables. This condition has been inspired by the guarded fragment of first-order logic, and is powerful enough to capture important Description Logics (DLs) such as the members of the $\mathcal{EL}$ family. The key model-theoretic property of the class of guarded TGDs, which explains its robust behaviour, is the existence of tree-like universal models [11].

Stickiness On the other hand, sticky sets of TGDs are powerful enough to model interesting statements that are inherently non-tree-like, and thus, not expressible via guarded TGDs. Such a statement, for example, consists of the TGDs

$$\forall x \forall y (R(x, y) \rightarrow \exists z R(y, z) \wedge P(z)) \qquad \forall x \forall y (P(x) \wedge P(y) \rightarrow S(x, y)),$$

which compute the cartesian product of an infinite unary relation, a useful modeling feature that, in DL parlance, is known as concept product [32].

The fact that the set of TGDs constructed in the undecidability proof of [18] is neither guarded nor sticky raised the following question: is the semi-oblivious/restricted

chase termination problem decidable for guarded or sticky sets of TGDs? The current state of affairs concerning this central question is as follows:

- For the semi-oblivious chase and guarded TGDs the problem is well-understood: it is 2ExpTime-complete in general, and ExpTime-complete for predicates of bounded arity [8]. The same paper [8] considered also linear TGDs, i.e., TGDs with only one body atom, which form a central subclass of guarded TGDs: the problem becomes PSpace-complete, and NLogSpace-complete for predicates of bounded arity. An alternative proof for linear TGDs, which relies on derivation trees, a notion that was originally introduced in the context of ontological query answering [3], has been recently proposed in [24].
- For the restricted chase and guarded TGDs, it has been recently shown, by exploiting Monadic-Second Order Logic over infinite trees of bounded degree, that the problem is decidable in elementary time assuming only one atom in the head, whereas the case of more than one atoms in the head remains a challenging open problem [19]. The case of linear TGDs with only one atom in the head has been explicitly considered in [24], where the decidability of the problem in question has been shown by relying on derivation trees.
- Finally, for the restricted chase and sticky sets of TGDs, it has been recently shown, by exploiting Büchi Automata, that the problem is decidable in elementary time assuming only one atom in the head, whereas the case of more than one atoms in the head remains a challenging open problem [19].

Towards completing the picture concerning the chase termination problem, in this work we concentrate on the semi-oblivious chase and sticky sets of TGDs, and provide precise complexity results: PSpace-complete in general, and NLogSpace-complete for predicates of bounded arity. Our results apply also to the oblivious chase that, although is not very useful for practical purposes, it is a relevant technical tool due to its simplicity; for a discussion on the usefulness of the oblivious chase see [11].

1.5 Summary of Contributions

Our results can be summarized as follows:

- In Section 4, we provide a semantic characterization of non-termination of the semi-oblivious chase under sticky sets of TGDs via the existence of "path-like" infinite chase derivations, which forms the basis for our decision procedure.
- By exploiting the above semantic characterization, we then provide, in Section 5, a syntactic characterization of semi-oblivious chase termination via a graph-based condition. To this end, we exploit a recent syntactic characterization from [8] of the termination of the semi-oblivious chase under linear TGDs.
- In Section 6, by using the graph-based syntactic characterization from the previous section, we establish the desired complexity upper bounds for our problem: PSpace in general, and NLogSpace for predicates of bounded arity. We finally establish matching lower bounds. The PSpace-hardness is obtained by simulating the behaviour of a polynomial space Turing machine, while the NLogSpace-hardness via a reduction from graph reachability.

2 Preliminaries

We consider the mutually disjoint countably infinite sets $\mathbf{C}$, $\mathbf{N}$, and $\mathbf{V}$ of *constants*, *(labeled) nulls*, and *variables*, respectively. We refer to constants, nulls and variables as *terms*. For an integer $n > 0$, we may write $[n]$ for the set $\{1, \ldots, n\}$.

Relational Databases A *schema* $\mathbf{S}$ is a finite set of relation symbols (or predicates) with associated arity. We write R/n to denote that R has arity $n \geq 0$; we may also write $\mathsf{ar}(R)$ for the integer n. A *(predicate) position* of $\mathbf{S}$ is a pair (R, i), where $R/n \in \mathbf{S}$ and $i \in [n]$, that essentially identifies the i-th argument of R. We write $\mathsf{pos}(\mathbf{S})$ for the set of positions of $\mathbf{S}$, that is, the set $\{(R, i) \mid R/n \in \mathbf{S} \text{ and } i \in [n]\}$. An *atom* over $\mathbf{S}$ is an expression of the form $R(\bar{t})$, where $R/n \in \mathbf{S}$ and $\bar{t}$ is an n-tuple of terms. A *fact* is an atom whose arguments consist only of constants. An *(atom) position* of $R(\bar{t})$ is a pair $(R(\bar{t}), i)$, where $i \in [\mathsf{ar}(R)]$, that essentially identifies the predicate position (R, i) in $R(\bar{t})$. We write $R(\bar{t})[i]$ for the term occurring at position $(R(\bar{t}), i)$. Moreover, for a variable x in $\bar{t}$, $\mathsf{pos}(R(\bar{t}), x)$ is the set of predicate positions $\{(R, i) \mid R(\bar{t})[i] = x\}$. We write $\mathsf{var}(R(\bar{t}))$ for the set of variables in $\bar{t}$. The notations $\mathsf{pos}(\cdot, x)$ and $\mathsf{var}(\cdot)$ extend to sets of atoms. An *instance* over $\mathbf{S}$ is a (possibly infinite) set of atoms over $\mathbf{S}$ with constants and nulls. A *database* over $\mathbf{S}$ is a finite set of facts over $\mathbf{S}$. The *active domain* of an instance I, denoted $\mathsf{dom}(I)$, is the set of terms in I.

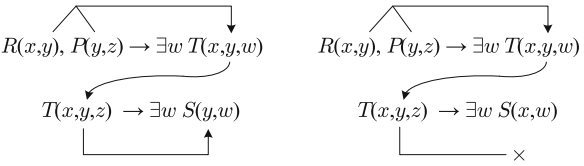
Substitutions and Homomorphisms A *substitution* from a set of terms T to a set of terms T' is a function $h : T \to T'$. Henceforth, we treat a substitution h as the set of mappings $\{t \mapsto h(t) \mid t \in T\}$. The restriction of h to a subset S of T, denoted $h_{|S}$, is the substitution $\{t \mapsto h(t) \mid t \in S\}$. A *homomorphism* from a set of atoms A to a set of atoms B is a substitution h from the set of terms in A to the set of terms in B such that (i) $t \in \mathbf{C}$ implies $h(t) = t$, i.e., h is the identity on $\mathbf{C}$, and (ii) $R(t_1, \ldots, t_n) \in A$ implies $h(R(t_1, \ldots, t_n)) = R(h(t_1), \ldots, h(t_n)) \in B$.

Tuple-Generating Dependencies A *tuple-generating dependency* (TGD) σ is a first-order sentence (without constants) of the form

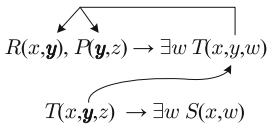
$$\forall \bar{x} \forall \bar{y} (\phi(\bar{x}, \bar{y}) \rightarrow \exists \bar{z} \psi(\bar{x}, \bar{z})),$$

where $\bar{x}$, $\bar{y}$ and $\bar{z}$ are mutually disjoint tuples of variables of $\mathbf{V}$, while $\phi(\bar{x}, \bar{y})$ and $\psi(\bar{x}, \bar{z})$ are conjunctions of atoms. For brevity, we write σ as $\phi(\bar{x}, \bar{y}) \rightarrow \exists \bar{z} \psi(\bar{x}, \bar{z})$, and use comma instead of $\wedge$ for joining atoms. We refer to $\phi(\bar{x}, \bar{y})$ and $\psi(\bar{x}, \bar{z})$ as the *body* and *head* of σ, denoted $\mathsf{body}(\sigma)$ and $\mathsf{head}(\sigma)$, respectively. The *frontier* of the TGD σ, denoted $\mathsf{fr}(\sigma)$, is the set of variables $\bar{x}$, i.e., the variables that appear both in the body and the head of σ. Note that, by abuse of notation, we may treat a tuple of variables as a set of variables. The *schema* of a set Σ of TGDs, denoted $\mathsf{sch}(\Sigma)$, is the set of predicates occurring in Σ, and we write $\mathsf{ar}(\Sigma)$ for the maximum arity over all those predicates. An instance I satisfies a TGD σ as the one above, written $I \models \sigma$, if the following holds: whenever there exists a homomorphism h from $\phi(\bar{x}, \bar{y})$ to I, then there exists $h' \supseteq h_{|\bar{x}}$ that is a homomorphism from $\psi(\bar{x}, \bar{z})$ to I. Note that, by abuse of notation, we sometimes treat a conjunction of atoms as a set of atoms. The instance I satisfies a set Σ of TGDs, written $I \models \Sigma$, if $I \models \sigma$ for each $\sigma \in \Sigma$.

Stickiness One of the main syntactic paradigms for TGDs is stickiness [13]. The key property underlying this condition is as follows: variables that appear more than once in the body of a TGD should be inductively propagated (or "stick") to every head atom, which can be graphically illustrated as



where the first set of TGDs is sticky, while the second is not. The formal definition relies on an inductive procedure that marks the variables that may violate the above property. The base step marks a variable in the body of a TGD that does not occur in every head atom. Then, the marking is inductively propagated as follows



Stickiness requires a marked variable to appear only once in the body of a TGD.

Let us now formalize the above intuitive discussion. Consider a set Σ of TGDs; we assume, w.l.o.g., that the TGDs in Σ do not share variables. Let $\sigma \in \Sigma$ and x a variable in $\mathsf{body}(\sigma)$. We inductively define when x *is marked in* Σ:

- If x does not occur in every atom of $\mathsf{head}(\sigma)$, then x is marked in Σ.
- Assuming that $\mathsf{head}(\sigma)$ contains an atom of the form $R(\bar{t})$ and $x \in \bar{t}$, if there exists $\sigma' \in \Sigma$ that has in its body an atom of the form $R(\bar{t}')$, and each variable in $R(\bar{t}')$ at a position of $\mathsf{pos}(R(\bar{t}), x)$ is marked in Σ, then x is marked in Σ.

The set Σ is *sticky* if there is no TGD whose body contains two occurrences of a variable that is marked in Σ. We denote by $\mathbb{S}$ the family of all sticky finite sets of TGDs.[1]

3 The Chase Procedure

The chase procedure (or simply chase) accepts as an input a database D and a set Σ of TGDs, and constructs a (possibly infinite) instance that contains D and satisfies Σ. Central notions in this context are those of active trigger and trigger application, which are coming into two different variations, oblivious and semi-oblivious, which in turn give rise to the oblivious [11] and the semi-oblivious [20, 28] chase. The key difference between these two versions of the chase, lies at the adopted naming scheme for the newly generated null values, which are used as witnesses for the existentially quantified variables occurring in the head of a TGD.

[1] We work with finite sets of TGDs only. Thus, in the rest of the paper, a set of TGDs is always finite.

Definition 1 (**Trigger and Trigger Application**) A *trigger* for a set Σ of TGDs on an instance I is a pair (σ, h), where $\sigma \in \Sigma$ and h is a homomorphism from $\mathsf{body}(\sigma)$ to I. The *oblivious result* and *semi-oblivious result* of (σ, h), denoted $\mathsf{o\text{-}result}(\sigma, h)$ and $\mathsf{so\text{-}result}(\sigma, h)$, is the set of atoms $\mu_{\mathsf{o}}(\mathsf{head}(\sigma))$ and $\mu_{\mathsf{so}}(\mathsf{head}(\sigma))$, respectively, where $\mu_{\mathsf{o}}, \mu_{\mathsf{so}} : \mathsf{var}(\mathsf{head}(\sigma)) \rightarrow \mathbf{C} \cup \mathbf{N}$ are defined as

$$\mu_{\mathsf{o}}(x) = \begin{cases} h(x) & \text{if } x \in \mathsf{fr}(\sigma) \\ \perp^{x}_{\sigma,h} & \text{otherwise} \end{cases} \qquad \mu_{\mathsf{so}}(x) = \begin{cases} h(x) & \text{if } x \in \mathsf{fr}(\sigma) \\ \perp^{x}_{\sigma,h_{|\mathsf{fr}(\sigma)}} & \text{otherwise} \end{cases}$$

where $\perp^{x}_{\sigma,h}$, $\perp^{x}_{\sigma,h_{|\mathsf{fr}(\sigma)}}$ are nulls from $\mathbf{N}$. We call the trigger (σ, h) *obliviously active* if $\mathsf{o\text{-}result}(\sigma, h) \not\subseteq I$, and *semi-obliviously active* if $\mathsf{so\text{-}result}(\sigma, h) \not\subseteq I$. The *oblivious application* of (σ, h) to I returns the instance $J = I \cup \mathsf{o\text{-}result}(\sigma, h)$, and is denoted as $I\langle \mathsf{o}, \sigma, h\rangle J$. Analogously, the *semi-oblivious application* of (σ, h) to I returns the instance $J = I \cup \mathsf{so\text{-}result}(\sigma, h)$, and is denoted as $I\langle \mathsf{so}, \sigma, h\rangle J$.

Observe that in the definition of $\star\text{-}\mathsf{result}(\sigma, h)$, where $\star \in \{\mathsf{o}, \mathsf{so}\}$, each existentially quantified variable x occurring in $\mathsf{head}(\sigma)$ is mapped by $\mu_\star$ to a "fresh" null value of $\mathbf{N}$ whose name is uniquely determined by the trigger (σ, h) and x itself. Therefore, for a trigger (σ, h), we can unambiguously write down the set of atoms $\star\text{-}\mathsf{result}(\sigma, h)$. In our analysis, it would be useful to be able to refer to the terms occurring in $\star\text{-}\mathsf{result}(\sigma, h)$ that have been propagated (not invented) during the application of (σ, h). Formally, the *frontier* of $\star\text{-}\mathsf{result}(\sigma, h)$, denoted $\mathsf{fr}(\star\text{-}\mathsf{result}(\sigma, h))$, is the set of terms $\mathsf{dom}(h(\mathsf{body}(\sigma))) \cap \mathsf{dom}(\star\text{-}\mathsf{result}(\sigma, h))$.

(Semi-)Oblivious Chase The main idea of the chase is, starting from a database D, to exhaustively apply active triggers for the given set Σ of TGDs on the instance constructed so far. We simultaneously define oblivious and semi-oblivious chase derivations. To this end, we distinguish the two cases where a derivation is finite or not:

- A finite sequence $(I_i)_{0 \le i \le n}$ of instances, with $D = I_0$ and $n \ge 0$, is an *oblivious* (resp., *semi-oblivious*) *chase derivation* of D w.r.t. Σ if, for each $i \in \{0, \ldots, n-1\}$, there exists an obliviously (resp., semi-obliviously) active trigger (σ, h) for Σ on I_i such that $I_i\langle \mathsf{o}, \sigma, h\rangle I_{i+1}$ (resp., $I_i\langle \mathsf{so}, \sigma, h\rangle I_{i+1}$), and no obliviously (resp., semi-obliviously) active trigger for Σ on I_n exists.
- An infinite sequence $(I_i)_{i \ge 0}$ of instances, with $D = I_0$, is an *oblivious* (resp., *semi-oblivious*) *chase derivation* of D w.r.t. Σ if, for each $i \ge 0$, there exists an obliviously (resp., semi-obliviously) active trigger (σ, h) for Σ on I_i such that $I_i\langle \mathsf{o}, \sigma, h\rangle I_{i+1}$ (resp., $I_i\langle \mathsf{so}, \sigma, h\rangle I_{i+1}$). Moreover, $(I_i)_{i \ge 0}$ is *fair* if, for each $i \ge 0$, and for every obliviously (resp., semi-obliviously) active trigger (σ, h) for Σ on I_i, there exists $j > i$ such that (σ, h) is not an obliviously (resp., semi-obliviously) active trigger for Σ on I_j. The latter is known as the *fairness condition*, and guarantees that all the active triggers will eventually be deactivated.

A (semi-)oblivious chase derivation is *valid* if it is finite, or infinite and fair. Infinite but unfair chase derivations are not valid since they do not serve the main purpose of the chase procedure, i.e., build an instance that satisfies the given TGDs. Henceforth, we write o-chase and so-chase for oblivious and semi-oblivious chase, respectively.

In general, due to the adopted naming scheme and the definition of active triggers, the semi-oblivious chase builds smaller instances than the oblivious one. This is because a trigger that is semi-obliviously active it is also obliviously active, but the other direction is not always true. This has been already illustrated by Example 2 in Section 1, which, for the sake of readability, we recall below.

Example 3 Consider the database $D = \{R(a, a)\}$, and the TGD

$$\sigma \;=\; R(x, y) \rightarrow \exists z R(x, z).$$

It is easy to verify that the only o-chase derivation of D w.r.t. $\{\sigma\}$ is infinite. On the other hand, the only so-chase derivation of D w.r.t. $\{\sigma\}$ is

$$\{R(a, a)\}, \left\{R(a, a), R\left(a, \perp^z_{\sigma,\{x\mapsto a\}}\right)\right\},$$

which is, of course, finite. Indeed, if we apply again the TGD σ, then we will obtain the atom $R\left(a, \perp^z_{\sigma,\{x\mapsto a\}}\right)$, which is already present.

Chase Relation A useful notion that we are going to use in our proofs is the so-called chase relation [13], which essentially describes how the atoms generated during the chase depend on each other. Consider a $\star$-chase derivation $\delta = (I_i)_{i\geq 0}$, where $\star \in \{\mathsf{o}, \mathsf{so}\}$, of a database D w.r.t. a set Σ of TGDs, and assume that for each $i \geq 0$, $I_i \langle \star, \sigma_i, h_i \rangle I_{i+1}$, which means that $I_{i+1} = I_i \cup \star\text{-}\mathsf{result}(\sigma_i, h_i)$. The *chase relation* of δ, denoted $\prec_\delta$, is a binary relation over $\bigcup_{i\geq 0} I_i$ such that $\alpha \prec_\delta \beta$ iff there exists $i \geq 0$ such that $\alpha \in h_i(\mathsf{body}(\sigma_i))$ and $\beta \in I_{i+1} \setminus I_i$. Notice that the relation $\prec_\delta$ is acyclic, or, in other words, it forms a directed acyclic graph over $\bigcup_{i\geq 0} I_i$.

3.1 Chase Termination Problem

It is known that due to the existentially quantified variables, a valid $\star$-chase derivation, where $\star \in \{\mathsf{o}, \mathsf{so}\}$, may be infinite. This is true even for very simple settings: it is easy to verify that the only $\star$-chase derivation of $D = \{R(a, b)\}$ w.r.t. the set Σ consisting of the single TGD $R(x, y)\rightarrow\exists z R(y, z)$ is infinite. The question that comes up is, given a set Σ of TGDs, can we check whether, for every database D, all or some valid (semi-)oblivious chase derivations of D w.r.t. Σ are finite? Before formalizing the above problem, let us recall two central classes of sets of TGDs:

$$\mathbb{CT}^\star_{\forall\forall} = \left\{ \Sigma \;\middle|\; \begin{array}{l} \text{for } \textit{every} \text{ database } D, \\ \textit{every} \text{ valid } \star\text{-chase derivation of } D \text{ w.r.t. } \Sigma \text{ is finite} \end{array} \right\}$$

$$\mathbb{CT}^\star_{\forall\exists} = \left\{ \Sigma \;\middle|\; \begin{array}{l} \text{for } \textit{every} \text{ database } D, \\ \text{there } \textit{exists} \text{ a finite } \star\text{-chase derivation of } D \text{ w.r.t. } \Sigma \end{array} \right\}$$

The problems tackled in this work are as follows, where $\mathbb{C}$ is a class of sets of TGDs:

PROBLEM :	$\mathsf{CT}^{\star}_{\forall\forall}(\mathbb{C})$
INPUT :	Set $\Sigma \in \mathbb{C}$ of TGDs.
QUESTION :	$\Sigma \in \mathbb{CT}^{\star}_{\forall\forall}$?

PROBLEM :	$\mathsf{CT}^{\star}_{\forall\exists}(\mathbb{C})$
INPUT :	Set $\Sigma \in \mathbb{C}$ of TGDs.
QUESTION :	$\Sigma \in \mathbb{CT}^{\star}_{\forall\exists}$?

It is well-known from [20] that the following holds:

$$\mathbb{CT}^{\mathsf{o}}_{\forall\forall} = \mathbb{CT}^{\mathsf{o}}_{\forall\exists} \subset \mathbb{CT}^{\mathsf{so}}_{\forall\forall} = \mathbb{CT}^{\mathsf{so}}_{\forall\exists}.$$

This immediately implies that, after fixing the version of the chase in question, i.e., oblivious or semi-oblivious, the above decision problems are equivalent. Henceforth, for a class $\mathbb{C}$ of sets of TGDs, we simply refer to the problem $\mathsf{CT}^{\star}_{\forall}(\mathbb{C})$, and we write $\mathbb{CT}^{\star}_{\forall}$ for the classes $\mathbb{CT}^{\star}_{\forall\forall}$ and $\mathbb{CT}^{\star}_{\forall\exists}$, where $\star \in \{\mathsf{o}, \mathsf{so}\}$.

We know that our main decision problems are, in general, undecidable. Assuming that $\mathbb{TGD}$ denotes the class of arbitrary sets of TGDs, we have that:

Theorem 1 ([18]) *For $\star \in \{\mathsf{o}, \mathsf{so}\}$, $\mathsf{CT}^{\star}_{\forall}(\mathbb{TGD})$ is undecidable, even if we focus on binary and ternary predicates.*

However, the set of TGDs employed in the undecidability proof of [18] is not sticky. What about $\mathsf{CT}^{\star}_{\forall}(\mathbb{S})$ then? This is a non-trivial problem, and pinpointing its complexity is the main goal of this work.

3.2 Some Useful Results

Before proceeding with the complexity analysis of $\mathsf{CT}^{\star}_{\forall}(\mathbb{S})$, let us recall a couple of technical results that would allow us to significantly simplify our investigation.

Critical Database It would be useful to have a special database of a very simple form that gives rise to a valid infinite chase derivation whenever there is a database with the same property. Interestingly, such a database exists, which is known as the critical database for a set Σ of TGDs [28]. Formally, the *critical database* for Σ is

$$\mathsf{cr}(\Sigma) = \{R(\natural, \ldots, \natural) \mid R \in \mathsf{sch}(\Sigma)\},$$

where $\natural \in \mathbf{C}$ is a fixed constant. In other words, $\mathsf{cr}(\Sigma)$ consists of all the atoms that can be formed using the predicates of $\mathsf{sch}(\Sigma)$ and the constant $\natural$. The following result states that $\mathsf{cr}(\Sigma)$ is indeed the desired database:

Proposition 1 ([28]) *Consider a set Σ of TGDs. For $\star \in \{\mathsf{o}, \mathsf{so}\}$, the following are equivalent:*

1. $\Sigma \notin \mathbb{CT}^{\star}_{\forall}$.
2. *There exists a valid infinite $\star$-chase derivation of $\mathsf{cr}(\Sigma)$ w.r.t. Σ.*

Henceforth, we always use $\natural$ for the constant that appears in a critical database. Notice that this special constant does not depend on the set of TGDs in question.

Fairness As one might expect, we are going to focus on the complement of $\mathsf{CT}^{\star}_{\forall}(\mathbb{S})$, for $\star \in \{\mathsf{o}, \mathsf{so}\}$, and pinpoint the complexity of the following problem: given a set $\Sigma \in \mathbb{S}$, is there a valid infinite $\star$-chase derivation δ of $\mathsf{cr}(\Sigma)$ w.r.t. Σ (see Proposition 1). However, as observed in [8], where the same problem is studied but for the class of guarded TGDs, one of the main difficulties is to ensure that δ enjoys the fairness condition. Interestingly, as shown in [8], we can completely neglect the fairness condition since the existence of a (possibly unfair) infinite $\star$-chase derivation of some database w.r.t. Σ implies the existence of a fair one.

Proposition 2 ([8]) *Consider a database D and a set Σ of TGDs. For $\star \in \{\mathsf{o}, \mathsf{so}\}$, the following are equivalent:*

1. *There exists a valid infinite $\star$-chase derivation of D w.r.t. Σ.*
2. *There exists an infinite $\star$-chase derivation of D w.r.t. Σ*

By combining Propositions 1 and 2, we obtain the following useful corollary:

Corollary 1 *Consider a set Σ of TGDs. For $\star \in \{\mathsf{o}, \mathsf{so}\}$, the following are equivalent*

1. $\Sigma \notin \mathbb{CT}^{\star}_{\forall}$.
2. *There exists an infinite $\star$-chase derivation of $\mathsf{cr}(\Sigma)$ w.r.t Σ.*

3.3 Our Main Result and Plan of Attack

As discussed above, the main goal of this work is to pinpoint the complexity of chase termination under sticky sets of TGDs, focussing on the oblivious and semi-oblivious versions of the chase procedure. Our main result follows:

Theorem 2 $\mathsf{CT}^{\mathsf{o}}_{\forall}(\mathbb{S})$ *and* $\mathsf{CT}^{\mathsf{so}}_{\forall}(\mathbb{S})$ *are* PSPACE*-complete, and* NLOGSPACE*-complete for predicates of bounded arity.*

Consider a set $\Sigma \in \mathbb{S}$. By Corollary 1, our main challenge is to show that the problem of deciding whether there exists an infinite $\star$-chase derivation of $\mathsf{cr}(\Sigma)$ w.r.t. Σ, where $\star \in \{\mathsf{o}, \mathsf{so}\}$, is PSPACE-complete, and NLOGSPACE-complete for predicates of bounded arity. In fact, the bulk of our work concentrates on establishing the desired upper bounds for the semi-oblivious chase, i.e., when $\star = \mathsf{so}$. We can then easily obtain the same upper bounds for the oblivious chase by exploiting a simple reduction from $\mathsf{CT}^{\mathsf{o}}_{\forall}(\mathbb{S})$ to $\mathsf{CT}^{\mathsf{so}}_{\forall}(\mathbb{S})$. Our plan of attack follows:

– The upper bounds heavily rely on the following semantic characterization given in Section 4: there exists an infinite so-chase derivation of $\mathsf{cr}(\Sigma)$ w.r.t. Σ iff there exists a "path-like" infinite so-chase derivation of $\mathsf{cr}(\Sigma)$ w.r.t. Σ.
– The above semantic characterization allows us to provide, in Section 5, a syntactic graph-based characterization of the existence of an infinite so-chase derivation of $\mathsf{cr}(\Sigma)$ w.r.t. Σ. Actually, the latter coincides with the existence of a certain "bad" cycle in the dependency graph of a "linearized" version of Σ.

– We show, in Section 6, that checking whether a "bad" cycle exists in the dependency graph of the "linearized" version of Σ is in PSPACE, and in NLOGSPACE for predicates of bounded arity. This shows the desired upper bounds for $\mathsf{CT}^{\mathsf{so}}_{\forall}(\mathbb{S})$. We then explain how the upper bounds for $\mathsf{CT}^{\mathsf{so}}_{\forall}(\mathbb{S})$ can be transferred to $\mathsf{CT}^{\mathsf{o}}_{\forall}(\mathbb{S})$ by exploiting a simple construction known as enrichment [20]. We finally provide matching lower bounds for $\mathsf{CT}^{\mathsf{o}}_{\forall}(\mathbb{S})$ and $\mathsf{CT}^{\mathsf{so}}_{\forall}(\mathbb{S})$. The PSPACE-hardness is obtained by simulating a deterministic polynomial space Turing machine, while the NLOGSPACE-hardness by a reduction from graph reachability.

At this point, one may wonder whether a powerful termination criterion from the literature allows us to characterize the fragment of sticky sets of TGDs that guarantees the termination of the semi-oblivious chase, which in turn will lead to the desired complexity results. To the best of our knowledge, such a criterion does not exist. Interestingly, *model-faithful acyclicity*, one of the largest classes of TGDs that ensure the termination of the semi-oblivious chase known today [14], is not powerful enough for characterizing the class $\mathbb{S} \cap \mathbb{CT}^{\mathsf{so}}_{\forall}$. For example, the sticky set of TGDs

$$P(x, y, z) \rightarrow S(x, y, z) \qquad S(x, y, x) \rightarrow \exists z P(y, z, x)$$

belongs to $\mathbb{CT}^{\mathsf{so}}_{\forall}$, but it violates the model-faithful acyclicity condition.

4 Semantic Characterization of Semi-Oblivious Chase Non-Termination

We proceed to characterize the non-termination of the so-chase under sticky sets of TGDs. For a set $\Sigma \in \mathbb{S}$, our goal is to show that, if there is an infinite so-chase derivation of $\mathsf{cr}(\Sigma)$ w.r.t. Σ, then we can isolate a "path-like" infinite so-chase derivation δ_ℓ, which we call *linear*. Roughly speaking, linearity means that there exists an infinite simple path $\alpha_0, \alpha_1, \alpha_2 \ldots$ in the chase relation of δ_ℓ such that $\alpha_0 \in \mathsf{cr}(\Sigma)$ and α_i is constructed during the i-th trigger application, while all the atoms that are needed to construct this path, and are not already on the path, are atoms of $\mathsf{cr}(\Sigma)$.

Definition 2 (**Linearity**) Consider a set Σ of TGDs. An infinite so-chase derivation $\delta = (I_i)_{i \geq 0}$ of $\mathsf{cr}(\Sigma)$ w.r.t. Σ, where $I_i \langle \mathsf{so}, \sigma_i, h_i \rangle I_{i+1}$ for $i \geq 0$, is called *linear* if there is an infinite sequence of distinct atoms $(\alpha_i)_{i \geq 0}$ such that the following hold:

1. $\alpha_0 \in \mathsf{cr}(\Sigma)$.
2. For each $i \geq 0$, $\alpha_{i+1} \in I_{i+1} \setminus I_i$, and there exists $\beta \in \mathsf{body}(\sigma_i)$ such that $h_i(\beta) = \alpha_i$ and $h_i(\mathsf{body}(\sigma_i) \setminus \{\beta\}) \subseteq \mathsf{cr}(\Sigma)$.

We are now ready to present our main characterization of non-termination of the semi-oblivious chase under sticky sets of TGDs.

Theorem 3 *Consider a set $\Sigma \in \mathbb{S}$. The following are equivalent:*

1. *There exists an infinite* so*-chase derivation of* $\mathsf{cr}(\Sigma)$ *w.r.t* Σ.

2. *There exists a linear infinite* so*-chase derivation of* $\mathsf{cr}(\Sigma)$ *w.r.t.* Σ.

It is clear that (2) $\Rightarrow$ (1) holds trivially. The non-trivial direction is (1) $\Rightarrow$ (2), which is established in two main steps:

1. We show that the chase relation of an infinite so-chase derivation δ of $\mathsf{cr}(\Sigma)$ w.r.t. Σ always contains a special path, called *continuous*, rooted at an atom of $\mathsf{cr}(\Sigma)$, which essentially guarantees the continuous propagation of a new null. Note that the existence of such a special path does not rely on stickiness.
2. By exploiting the existence of a continuous path, we construct a linear infinite so-chase derivation of $\mathsf{cr}(\Sigma)$ w.r.t. Σ. In fact, due to stickiness, we can convert an infinite suffix P of the continuous path in $\prec_\delta$, together with all the atoms that are needed to generate the atoms on P via a single trigger application, into a linear infinite so-chase derivation δ_ℓ of $\mathsf{cr}(\Sigma)$ w.r.t. Σ. As we shall see, stickiness helps us to ensure that δ_ℓ is linear, while continuity allows us to show that δ_ℓ is infinite.

In the rest of this section, we fix a set $\Sigma \in \mathbb{S}$. For technical clarity, we assume that all the TGDs of Σ have a non-empty frontier, i.e., for every TGD $\sigma \in \Sigma$, there exists at least one variable that appears both in $\mathsf{body}(\sigma)$ and $\mathsf{head}(\sigma)$. Furthermore, we assume that Σ is in *normal form*, i.e., each TGD of Σ has only one atom in its head. The normalization procedure, which preserves stickiness, is rather standard and can be found, e.g., in [13]. The above simplifying assumptions do not affect the generality of our proof. In other words, assuming that Σ is what we obtain after removing from $\hat{\Sigma} \in \mathbb{S}$ all the TGDs with an empty frontier and then normalize it, we can easily show that there exists a linear infinite so-chase derivation of $\mathsf{cr}(\hat{\Sigma})$ w.r.t. $\hat{\Sigma}$ iff there exists a linear infinite so-chase derivation of $\mathsf{cr}(\Sigma)$ w.r.t. Σ.

4.1 Existence of a Continuous Path

Let us first formalize the notion of path in the chase relation of a derivation. Given an infinite so-chase derivation $\delta = (I_i)_{i\geq 0}$ of $\mathsf{cr}(\Sigma)$ w.r.t. Σ, a *finite* δ*-path* is a finite sequence of atoms $(\alpha_i)_{0\leq i\leq n}$ such that $\alpha_0 \in I_0$ and $\alpha_i \prec_\delta \alpha_{i+1}$. Analogously, we can define the notion of *infinite* δ*-path*, which refers to an infinite sequence of atoms rooted at an atom of I_0. The intention underlying continuity is to ensure the continuous propagation of a new null on a path. Roughly, a δ-path $(\alpha_i)_{i\geq 0}$ is continuous via a sequence of indices $(\ell_i)_{i\geq 0}$, with $\ell_0 = 1$, if for each $i \geq 0$, a new null is invented in α_{ℓ_i} that is "necessarily propagated" up to the atom $\alpha_{\ell_{i+1}}$. At this point, it is crucial to formalize what we mean by "necessarily propagated".

Let α, β be atoms of $\bigcup_{i\geq 0} I_i$, and assume that $\beta \in I_j \setminus I_{j-1}$, for some $j > 0$, with $I_{j-1}\langle \mathsf{so}, \sigma, h\rangle I_j$, i.e., $\beta = \mathsf{so\text{-}result}(\sigma, h)$.[2] We say that the i-th position of α and the j-th position of β are *related (w.r.t* δ*)*, denoted $(\alpha, i) \simeq_\delta (\beta, j)$, if there exists an atom $\gamma \in \mathsf{body}(\sigma)$ such that $\alpha = h(\gamma)$ and $\gamma[i] = \mathsf{head}(\sigma)[j]$. A finite δ-path $(\alpha_i)_{0\leq i\leq n}$ is (s, t)*-connected*, where $s \in [\mathsf{ar}(R_0)]$ and $t \in [\mathsf{ar}(R_n)]$, with R_0 and R_n

[2]Recall that Σ is in normal form, which means that $\mathsf{head}(\sigma)$ is a single atom. By abuse of notation, we treat $\mathsf{so\text{-}result}(\sigma, h)$ as an atom instead of a singleton set.

being the predicates of α_0 and α_n, respectively, if there exists a sequence of integers $(\ell_k)_{1\leq k\leq n-1}$ such that

$$(\alpha_0, s) \simeq_\delta (\alpha_1, \ell_1) \simeq_\delta (\alpha_2, \ell_2) \simeq_\delta \cdots \simeq_\delta (\alpha_{n-1}, \ell_{n-1}) \simeq_\delta (\alpha_n, t).$$

In simple words, the term occurring at position (α_0, s) is necessarily propagated up to the position (α_n, t) via the intermediate positions $(\alpha_1, \ell_1), \ldots, (\alpha_{n-1}, \ell_{n-1})$.

For introducing continuity, we also need the notion of the birth atom of a null value. Consider a null $\bot$ occurring in $\bigcup_{i\geq 0} I_i$. The *birth atom (w.r.t. δ)* of $\bot$, denoted $\mathsf{birth}_\delta(\bot)$, is the atom of $\bigcup_{i\geq 0} I_i$ such that, for some $j > 0$, $I_j \setminus I_{j-1} = \{\mathsf{birth}_\delta(\bot)\}$ and $\bot \in \mathsf{dom}(I_j) \setminus \mathsf{dom}(I_{j-1})$. It is clear that the atom $\mathsf{birth}_\delta(\bot)$ is unique (since we consider TGDs in normal form). We are now ready to introduce continuity.

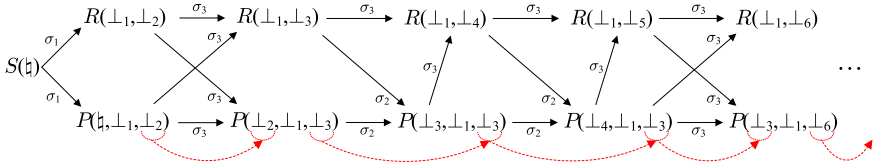
Definition 3 (**Continuity**) Let δ be an infinite so-chase derivation δ of $\mathsf{cr}(\Sigma)$ w.r.t. Σ. A δ-path $(\alpha_i)_{i\geq 0}$ is *continuous* if there exist an infinite sequence $(\ell_i)_{i\geq 0}$ of integers, where $\ell_0 = 1$ and $\ell_0 < \ell_1 < \cdots$, an infinite sequence $(\bot_i)_{i\geq 0}$ of nulls, and infinite sequences of integers $(m_i)_{i\geq 0}$ and $(p_i)_{i\geq 0}$ from $[\mathsf{ar}(\Sigma)]$, such that, for each $k \geq 0$: $\alpha_{\ell_k} = \mathsf{birth}_\delta(\bot_k)$, $\bot_k = \alpha_{\ell_k}[m_k]$, and $(\alpha_i)_{\ell_k\leq i\leq \ell_{k+1}}$ is (m_k, p_k)-connected.

A simple example that illustrates the notion of continuity follows:

Example 4 Assume that $\Sigma = \{\sigma_1, \sigma_2, \sigma_3\}$, where[3]

$$\begin{aligned}
\sigma_1 &= S(x) \rightarrow \exists y \exists z P(x, y, z), R(y, z)\\
\sigma_2 &= P(x, y, z), R(y, w) \rightarrow P(w, y, z)\\
\sigma_3 &= P(x, y, z), R(y, w) \rightarrow \exists v P(z, y, v), R(y, v), Q(w).
\end{aligned}$$

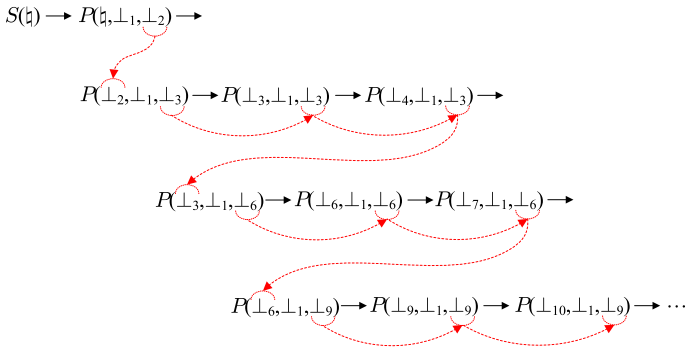
It is easy to verify that there exists an infinite so-chase derivation δ of $\mathsf{cr}(\Sigma)$ w.r.t. Σ such that the following is part of $\prec_\delta$; a black solid edge from α to β labeled by σ means that (α, β) belongs to $\prec_\delta$ due to a trigger that involves the TGD σ:



It can be verified that the path with P-atoms is a continuous δ-path. Let us explain the reason. The first atom in which a null is born is $P(c, \bot_1, \bot_2)$, with $\bot_1, \bot_2$ being the new nulls, and continuity is satisfied since $\bot_2$ is propagated (this is indicated via the red dashed arrows) to the next atom where the new null $\bot_3$ is born. Now, since the null $\bot_3$ is propagated up to the next birth atom $P(\bot_3, \bot_1, \bot_6)$, continuity

[3]Although we assume that Σ is in normal form, for the sake of readability, in our example we use TGDs with more than one atom in the head. Moreover, instead of following the naming scheme for nulls introduced in Definition 1, we are going to use abbreviated names of the form $\bot_i$, where $i > 0$.

is satisfied. In the rest of the path the same pattern is repeated, and thus continuity is globally satisfied. In fact, the pattern that we can extract is graphically illustrated as



where the continuous propagation of a new null (red dashed arrows) can be seen.

We are now ready to establish our main technical result concerning continuity. Note that the next result holds for arbitrary (not necessarily sticky) sets of TGDs.

Lemma 1 *For every infinite* so*-chase derivation* δ *of* $\mathsf{cr}(\Sigma)$ *w.r.t* Σ*, there exists a* δ*-path that is continuous.*

Proof Assume that $\delta = (I_i)_{i \geq 0}$ is an infinite so-chase derivation of $\mathsf{cr}(\Sigma)$ w.r.t Σ, and let $\mathcal{I} = \bigcup_{i \geq 0} I_i$. Given a term $t \in \mathsf{dom}(\mathcal{I})$, which is either the constant $\natural$ or a null, and a null $\bot \in \mathsf{dom}(\mathcal{I})$, we write $t \rhd_\delta \bot$ if t occurs in $\mathsf{fr}(\mathsf{birth}_\delta(\bot))$.[4] Now, for each $t \in \mathsf{dom}(\mathcal{I})$, we inductively define the rank (w.r.t. δ) of t as follows:

$$\mathsf{rank}_\delta(t) = \begin{cases} 0 & \text{if } t = \natural, \\ 1 + \max\{\mathsf{rank}(t') \mid t' \rhd_\delta t\} & \text{otherwise.} \end{cases}$$

For a null $\bot \in \mathsf{dom}(\mathcal{I})$, we select a term $t \in \mathsf{dom}(\mathcal{I})$ such that $\mathsf{rank}(t) = \mathsf{rank}(\bot) - 1$ and $t \rhd_\delta \bot$, and we write $t \blacktriangleright_\delta \bot$.[5] Since, by assumption, all the TGDs of Σ have a non-empty frontier, it is easy to see that the binary relation $\blacktriangleright_\delta$ over $\mathsf{dom}(\mathcal{I})$ forms an infinite rooted tree T_δ, where the root is the constant occurring in $\mathsf{cr}(\Sigma)$. The key property of T_δ is given by the following result:

Lemma 2 *T_δ has finite out-degree.*

Proof We can show that, for each $i \geq 0$, the set $\{t \in \mathsf{dom}(\mathcal{I}) \mid \mathsf{rank}(t) = i\}$ is finite. This can be shown by induction on $i \geq 0$, while the key fact is that only finitely many

[4] Since we focus on TGDs with only one head atom, and since we apply only active triggers, given an atom $\alpha \in \mathcal{I} \setminus \mathsf{cr}(\Sigma)$ that is the birth atom of a null, we can extract *the* trigger that generates α, and therefore, we can unambiguously refer to its frontier.

[5] We assume that there exists some fixed mechanism that selects t. For example, t can be the lexicographically first element of $\{t' \mid \mathsf{rank}_\delta(t') = \mathsf{rank}(\bot) - 1 \text{ and } t' \rhd_\delta \bot\}$.

semi-obliviously active triggers can be formed due to which a null with rank $i+1$ is generated (since, by induction hypothesis, the set of terms with rank at most i is finite). Therefore, the nodes of T_δ have finite out-degree. □

Having Lemma 2 in place, we can apply König's Lemma on T_δ, and get that T_δ contains an infinite simple path starting from the root node.[6] Let $\perp_0, \perp_1, \perp_2, \ldots$ be such a path, where $\perp_0 = \natural$. By construction, for each $i \geq 0$, $\perp_i \in \mathsf{fr}(\mathsf{birth}_\delta(\perp_{i+1}))$. Moreover, there is a sequence of atoms

$$\alpha^i_0, \alpha^i_1, \alpha^i_2, \ldots, \alpha^i_{m_i},$$

where $\alpha^i_0 \in \mathsf{cr}(\Sigma)$ if $i = 0$, $\alpha^i_0 = \mathsf{birth}_\delta(\perp_i)$ if $i > 0$, and $\alpha^i_{m_i} = \mathsf{birth}_\delta(\perp_{i+1})$, such that $\alpha^i_k \prec_\delta \alpha^i_{k+1}$, for each $k \in \{0, \ldots, m_i - 1\}$, and there are integers $s, t \in [\mathsf{ar}(\Sigma)]$ such that $(\alpha^i_j)_{0 \leq j \leq m_i}$ is (s,t)-connected. Consequently, the infinite sequence

$$\alpha^0_0, \alpha^0_1 = \alpha^1_0, \alpha^1_1, \ldots, \alpha^1_{m_1} = \alpha^2_0, \alpha^2_1, \ldots, \alpha^2_{m_2} = \alpha^3_0, \ldots$$

is a continuous δ-path, and the claim follows. □

4.2 From Arbitrary to Linear Infinite Derivations

We can now show that an infinite so-chase derivation $\delta = (I_i)_{i \geq 0}$ of $\mathsf{cr}(\Sigma)$ w.r.t. Σ can always be converted into a linear infinite so-chase derivation δ_ℓ, which in turn establishes the non-trivial direction (1) $\Rightarrow$ (2) of Theorem 3. The construction proceeds in three main steps, which are described below. We first describe those steps in a semi-formal way, and exploit Example 4 in order to illustrate the key ideas. We then proceed to give the formal construction.

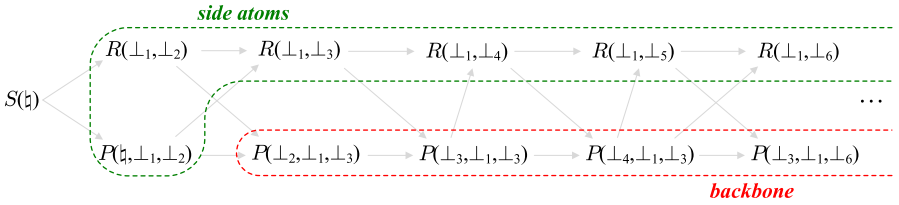
4.2.1 Semi-Formal Description

Useful part of δ We first isolate a useful part of the so-chase derivation $\delta = (I_i)_{i \geq 0}$. By Lemma 1, there exists a continuous infinite δ-path $P = (\alpha_i)_{i \geq 0}$. Recall that continuity guarantees the continuous propagation on P of infinitely many nulls, which we call for the purpose of this discussion *pivotal*. By stickiness, there exists $j \geq 0$ such that α_j is the last atom on P in which a term t becomes *sticky*. By saying that the term t becomes sticky, we mean that the first time t participates in a join is during the trigger application that generates α_j, and thus t occurs in (or sticks to) every atom of $\{\alpha_i\}_{i \geq j}$. Let $k \geq j$ be the integer such that α_k is the first atom on P after α_j in which a new pivotal null is invented. The useful part of δ that we are going to focus on is the infinite sequence of atoms $(\alpha_i)_{i \geq k}$, which we call the *backbone*, and the atoms of $\bigcup_{i \geq 0} I_i$, which we call *side atoms*, that are needed to generate the atoms

[6] König's Lemma is a well-known result from graph theory, which states that an infinite rooted tree with finite out-degree has an infinite directed simple path starting from the root.

on the backbone via a single trigger application. In other words, for a backbone atom α, if α is obtained via the trigger (σ, h) for Σ on instance I_i, for some $i \geq 0$, then the atoms $h(\mathsf{body}(\sigma))$, excluding the backbone atoms, are side atoms.

Example 5 Consider again the set $\Sigma \in \mathbb{S}$ from Example 4. As discussed above, there exists an infinite so-chase derivation δ of $\mathsf{cr}(\Sigma)$ w.r.t. Σ such that a continuous infinite δ-path exists (see the figures above). The useful part of δ is as shown below



Observe that the last atom on the continuous path in which a term becomes sticky is $P(\bot_2, \bot_1, \bot_3)$; in fact, the sticky term is $\bot_1$, which is the only sticky term on the continuous path. It happened that $P(\bot_2, \bot_1, \bot_3)$ invents also a new pivotal null, that is, $\bot_3$, and therefore the suffix of the continuous path that starts at $P(\bot_2, \bot_1, \bot_3)$ is the backbone. It is now easy to verify that all the other atoms, apart from $S(\natural)$, indeed contribute to the generation of a backbone atom via a single trigger application.

Renaming step We proceed to rename some of the nulls that occur in backbone atoms or side atoms. In particular, for every null $\bot$ occurring in a side atom α, we apply the following renaming steps; recall that $\natural$ is the constant occurring in $\mathsf{cr}(\Sigma)$: (i) every occurrence of $\bot$ in α is replaced by $\natural$, and (ii) every occurrence of $\bot$ in a backbone atom β that is propagated from α to β is replaced by $\natural$. For a backbone or side atom α, let $\rho(\alpha)$ be the atom obtained from α after globally applying the above renaming steps. We now define the sequence of instances $\delta' = (J_i)_{i \geq 0}$ as follows:

$$J_0 \;=\; \{\rho(\alpha) \mid \alpha \text{ is a side atom}\},$$

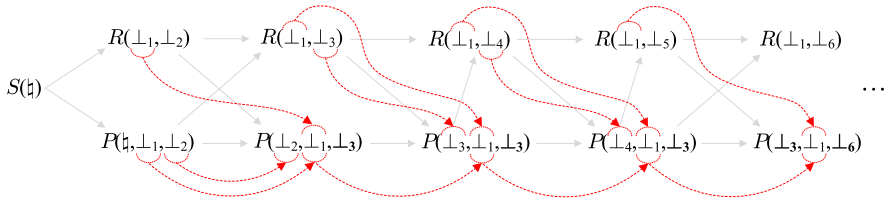
which is a subset of $\mathsf{cr}(\Sigma)$, and

$$J_i \;=\; J_{i-1} \cup \{\rho(\alpha_{k+i-1})\} \cup H, \quad \text{for } i > 0,$$

where H is the set of atoms that are generated together with α_{k+i-1}, after renaming the propagated nulls that do not occur in $\rho(\alpha_{k+i-1})$ to $\natural$.[7] It is crucial to observe that a new null generated in a backbone atom never participates in a join. This is because the first backbone atom α_k comes after the atom α_j, which is the last atom on P in which a term becomes sticky. This fact allows us to modify triggers from δ in order to construct, for every $i \geq 0$, a trigger (σ_i, h_i) such that $J_i \langle \mathsf{so}, \sigma_i, h_i \rangle J_{i+1}$.

[7]Recall that, although we assume TGDs in normal form, for the sake of readability, in Example 4 (and thus, also in Example 5), we use TGDs with more than one atom in the head. This is why the set of atoms H is considered. Notice that H is empty in case of single-head TGDs.

Example 6 We consider again our running example. Before renaming the nulls that appear in side atoms, we first need to understand how nulls are propagated from side atoms to backbone atoms during the chase. This is depicted in the following figure



Note that the boldfaced occurrences of the pivotal nulls $\bot_3, \bot_6, \ldots$ are not propagated from side atoms, but generated on the backbone, and thus will not be renamed. Recall that the existence of such nulls is guaranteed by continuity. By applying the renaming step, i.e., by replacing every null in a side atom with the constant $\natural$, and then propagating it to the backbone as indicated above, we get the sequence of instances

$$\begin{aligned} J_0 &= \{R(\natural, \natural), P(\natural, \natural, \natural)\}, \\ J_1 &= J_0 \cup \{P(\natural, \natural, \bot_3), R(\natural, \bot_3)\}, \\ J_2 &= J_1 \cup \{P(\natural, \natural, \bot_3)\}, \\ J_3 &= J_2 \cup \{P(\natural, \natural, \bot_3)\}, \\ J_4 &= J_3 \cup \{P(\bot_3, \natural, \bot_6), R(\natural, \bot_6)\}, \\ &\vdots \end{aligned}$$

Observe that, due to stickiness, none of the nulls $\bot_3, \bot_6, \ldots$ generated on the backbone participates in a join. Hence, the renaming step preserves all the joins, and thus, by adapting triggers from δ, we can devise a trigger for each pair (J_i, J_{i+1}).

Pruning step At this point, one may be tempted to think that $\delta' = (J_i)_{i \geq 0}$, with $J_i \langle \mathsf{so}, \sigma_i, h_i \rangle J_{i+1}$ for $i \geq 0$, is the desired linear infinite so-chase derivation of $\mathsf{cr}(\Sigma)$ w.r.t. Σ. It is easy to verify that we have the infinite sequence of atoms $(\rho(\alpha_i))_{i \geq k-1}$ such that $\rho(\alpha_{k-1}) \in \mathsf{cr}(\Sigma)$ since α_{k-1} is a side atom, and for each $i \geq k-1$, $J_{i-k+2} \supseteq J_{i-k+1} \cup \{\rho(\alpha_{i+1})\}$, and there exists $\beta \in \mathsf{body}(\sigma_i)$ such that $h_i(\beta_i) = \alpha_i$ and $h_i(\mathsf{body}(\sigma_i) \setminus \{\beta\}) \subseteq \mathsf{cr}(\Sigma)$. However, we cannot conclude yet that δ' is the desired so-chase derivation since we may apply a trigger that is not semi-obliviously active. This can be fixed by simply removing the instances that are obtained via a trigger that is not semi-obliviously active, which actually means that we keep only one occurrence of each instance. However, such a pruning step might be applied infinitely many times, and thus, the question that we need to answer is whether the obtained so-chase derivation δ'' remains infinite. Interestingly, this is the case due to continuity. Since the backbone $(\alpha_i)_{i \geq k}$ is part of a continuous δ-path, we get that all the (necessarily consecutive) occurrences of an instance appear between two instances in which new pivotal nulls are invented. Since the distance between such instances is finite,

each instance occurs in δ' finitely many times. Therefore, after the pruning step, the obtained so-chase derivation is infinite. Thus, δ'' is a linear infinite so-chase derivation of J_0 w.r.t. Σ. Since $J_0 \subseteq \mathsf{cr}(\Sigma)$, we can construct a linear infinite so-chase derivation δ_ℓ of $\mathsf{cr}(\Sigma)$ w.r.t. Σ by adding to J_0 the set of atoms $\mathsf{cr}(\Sigma) \setminus J_0$.

Example 7 Coming back to our running example, it can be seen that the sequence of instances devised in Example 6 is not the desired linear derivation due to non-active triggers, which implies that there are consecutive occurrences of the same instance, e.g., J_1, J_2, J_3. However, after applying the pruning step, we get the sequence

$$\begin{aligned} J'_0 &= J_0, \\ J'_1 &= J_1, \\ J'_2 &= J'_1 \cup \{P(\bot_3, \natural, \bot_6), R(\natural, \bot_6)\}, \\ J'_3 &= J'_2 \cup \{P(\bot_6, \natural, \bot_9), R(\natural, \bot_9)\}, \\ &\vdots \end{aligned}$$

Now, it is easy to verify that after adding the atoms $S(\natural)$ and $Q(\natural)$ to J'_0, we get a linear infinite so-chase derivation of $\mathsf{cr}(\Sigma)$ w.r.t. Σ.

4.2.2 The Formal Construction

Useful part of δ In what follows, let $\mathcal{I} = \bigcup_{i \geq 0} I_i$. By Lemma 1, there exists a continuous infinite δ-path $P = (\alpha_i)_{i \geq 0}$. That is, there exist an infinite sequence $(\ell_i)_{i \geq 0}$ of integers, where $\ell_0 = 1$ and $\ell_0 < \ell_1 < \cdots$, an infinite sequence $(\bot_i)_{i \geq 0}$ of nulls, and infinite sequences of integers $(m_i)_{i \geq 0}$ and $(p_i)_{i \geq 0}$ from $[\mathsf{ar}(\Sigma)]$, such that, for each $k \geq 0$, $\alpha_{\ell_k} = \mathsf{birth}_\delta(\bot_k)$, $\bot_k = \alpha_{\ell_k}[m_k]$, and $(\alpha_i)_{\ell_k \leq i \leq \ell_{k+1}}$ is (m_k, p_k)-connected. Given an atom $\alpha_i = \mathsf{so\text{-}result}(\sigma, h)$ for some $i > 0$, we say that a term $t \in \mathsf{dom}(\mathcal{I})$ *becomes sticky in* α_i if i is the smallest integer such that there is a variable $x \in \mathsf{fr}(\sigma)$ occurring more than once in $\mathsf{body}(\sigma)$ and $t = h(x)$. By stickiness of Σ, and from the fact that the arity of each predicate in Σ is finite, there exists some $j \geq 0$ such that, for every $i \geq j$, no term $t \in \mathsf{dom}(\mathcal{I})$ becomes sticky in α_i. Let $k \geq 0$ be the smallest integer such that no term becomes sticky in α_{ℓ_k}. Note that this integer exists since the sequence of indices $(\ell_i)_{i \geq 0}$ is infinite. We call the sequence of atoms $B = (\alpha_i)_{i \geq \ell_k}$ the *backbone of* δ; let $\mathcal{B} = \{\alpha_i\}_{i \geq \ell_k}$. Furthermore, we define the set

$$S = \{\alpha \in \mathcal{I} \setminus \mathcal{B} \mid \text{there exists } \beta \in \mathcal{B} \text{ such that } \alpha \prec_\delta \beta\},$$

which we call the *side atoms of* B. The set $S \cup \mathcal{B}$ is the *useful part of* δ.

Renaming step For notational convenience, let $B = (\beta_i)_{i \geq 0}$. We define B' as the sequence of atoms obtained from B as follows. For every atom $\beta = R(\bar{t}) \in \mathcal{B}$, and every integer $j \in [\mathsf{ar}(R)]$, if there exists an atom $\alpha = S(\bar{u}) \in S$, and an integer $i \in [\mathsf{ar}(S)]$, with $\alpha[i] \in \mathbf{N}$, such that an (i, j)-connected δ-path of the form $\alpha, \ldots, \beta$

exists, then the term at position (β, j) is renamed to the constant $\natural$. Furthermore, for each $i \geq 0$, assuming that $\beta_i = \mathsf{so\text{-}result}(\sigma, h)$, let $\eta_i = (\sigma, h')$ be the trigger obtained from (σ, h) after updating h to h' as dictated by the renaming step applied to β_i. Assuming $B' = (\beta'_i)_{i \geq 0}$, we define $\delta' = (J_i)_{i \geq 0}$ such that $J_0 = \mathsf{cr}(\Sigma)$, and for $i > 0$, $J_i = J_{i-1} \cup \{\beta'_{i-1}\}$. Since

- no term becomes sticky in some atom of B, and
- for each null $\perp$ such that $\beta = \mathsf{birth}_\delta(\perp) \in \mathcal{B}$, which means that $\perp$ has been generated on B, $\perp$ is not renamed on B',

we get that, for $i \geq 0$, η_i is a trigger for Σ on J_i, and thus, $J_{i+1} = J_i \cup \mathsf{so\text{-}result}(\eta_i)$.

Pruning step Although $\delta' = (J_i)_{i \geq 0}$ is *locally consistent*, that is, for each $i \geq 0$, $J_i \langle \mathsf{so}, \sigma_i, h_i \rangle J_{i+1}$, for some trigger (σ_i, h_i) for Σ on J_i, it is not yet a so-chase derivation since some of the involved triggers might not be semi-obliviously active. In other words, there might exist $i \geq 0$ such that $J_i \langle \mathsf{so}, \sigma_i, h_i \rangle J_{i+1}$, but $\mathsf{so\text{-}result}(\sigma_i, h_i) \in J_i$, which implies that $J_i = J_{i+1}$. Let δ'' be the sequence of instances obtained from δ' where only one occurrence of each instance is kept. Clearly, δ'' is still locally consistent. Moreover, by exploiting continuity, we can show that, for each $i \geq 0$, J_i occurs in δ' finitely many times, which in turn implies that δ'' is a linear infinite so-chase derivation of $\mathsf{cr}(\Sigma)$ w.r.t. Σ, as needed.

5 Graph-Based Characterization of Semi-Oblivious Chase Termination

In this section, we characterize the termination of the semi-oblivious chase for sticky sets of TGDs via a graph-based condition. More precisely, we show that a set $\Sigma \in \mathbb{S}$ belongs to $\mathbb{CT}^{\mathsf{so}}_\forall$ iff a linearized version of it, i.e., a set of linear TGD obtained from Σ, enjoys a condition inspired by weak-acyclicity [17], called critical-weak-acyclicity, introduced in [8]. Recall that linear TGDs are TGDs with only one body atom [12]; we write $\mathbb{L}$ for the class of linear TGDs. The proof of the above result boils down to showing that the given sticky set Σ of TGDs can be rewritten into a set of linear TGDs, while this rewriting preserves the termination of the semi-oblivious chase. The latter heavily relies on Theorem 3, which establishes that non-termination of the semi-oblivious chase coincides with the existence of a linear infinite chase derivation of $\mathsf{cr}(\Sigma)$ w.r.t. Σ. We can then apply the characterization for the termination of the semi-oblivious chase for linear TGDs from [8], which states that a set $\Sigma \in \mathbb{L}$ belongs to $\mathbb{CT}^{\mathsf{so}}_\forall$ iff it is critically-weakly-acyclic. Let us first recall the notion of critical-weak-acyclicity for linear TGDs, which has been originally introduced in [8].

5.1 Critically-Weakly-Acyclic Linear TGDs

Since critical-weak-acyclicity is inspired by weak-acyclicity, it is not surprising that it relies on the dependency graph of a set of TGDs introduced in [17], that encodes

how terms might be propagated during the chase. We assume a fixed ordering on the head-atoms of TGDs.[8] For a TGD σ with $\mathsf{head}(\sigma) = \alpha_1, \ldots, \alpha_k$, we write (σ, i) for the TGD that has only one atom in its head, called *single-head*, obtained from σ by keeping only the atom α_i, and the existentially quantified variables in α_i. Recall that $\mathsf{pos}(\alpha, x)$ is the set of positions in α at which x occurs, while $\mathsf{pos}(\mathsf{body}(\sigma), x)$ is the set of positions at which x occurs in the body of σ. We also write $\mathsf{pos}(\mathsf{sch}(\Sigma))$ for the set of positions of $\mathsf{sch}(\Sigma)$, i.e., the set $\{(R, i) \mid R/n \in \mathsf{sch}(\Sigma) \text{ and } i \in [n]\}$.

Definition 4 (**Dependency Graph**) The *dependency graph* of a set Σ of TGDs is a labeled directed multigraph $\mathsf{dg}(\Sigma) = (N, E, \lambda)$, where $N = \mathsf{pos}(\mathsf{sch}(\Sigma))$, $\lambda : E \to \Sigma \times \mathbb{N}$, and E contains only the following edges. For each $\sigma \in \Sigma$ with $\mathsf{head}(\sigma) = \alpha_1, \ldots, \alpha_k$, for each $x \in \mathsf{fr}(\sigma)$, and for each position $\pi \in \mathsf{pos}(\mathsf{body}(\sigma), x)$:

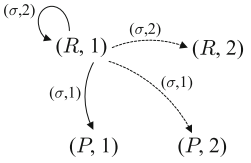
- For each $i \in [k]$, and for each position $\pi' \in \mathsf{pos}(\alpha_i, x)$, there is a *normal* edge $e = (\pi, \pi') \in E$ with $\lambda(e) = (\sigma, i)$.
- For each existentially quantified variable z in σ, for each $i \in [k]$, and for each $\pi' \in \mathsf{pos}(\alpha_i, z)$, there is a *special* edge $e = (\pi, \pi') \in E$ with $\lambda(e) = (\sigma, i)$.

Intuitively speaking, a normal edge (π, π') encodes the fact that a term may propagate from π to π' during the chase. Moreover, a special edge (π, π'') keeps track of the fact that the propagation of a term from π to π' also creates a new null at position π''. A simple example that illustrates the notion of the dependency graph follows:

Example 8 Consider the set Σ consisting of the TGD

$$\sigma = R(x, y) \to \exists z P(x, z), R(x, z).$$

The graph $\mathsf{dg}(\Sigma)$ is as follows



where the dashed arrows represent special edges. The normal edges occur due to the variable x, while the special edges due to the existentially quantified variable z.

The class of weakly-acyclic sets of TGDs is a well-known formalism, introduced in the context of data exchange, that guarantees the termination of the semi-oblivious chase [17].[9] Formally, a set Σ is *weakly-acyclic* if there is no cycle in $\mathsf{dg}(\Sigma)$ that

[8] Let us clarify that in this section, unlike Section 4, we consider arbitrary TGDs not in normal form. In the previous section, we assumed TGDs in normal form only for technical reasons.

[9] Let us clarify that [17] does not consider the semi-oblivious, but the restricted (a.k.a. the standard) version of the chase. However, the exact same proof applies in the case of the semi-oblivious chase.

contains a special edge. It would be extremely useful if, whenever we focus on linear TGDs, weak-acyclicity is also a necessary condition for the termination of the semi-oblivious chase. Unfortunately, this is not the case. A simple counterexample follows:

Example 9 Consider the set Σ of linear TGDs consisting of $R(x, x) \rightarrow \exists z R(z, x)$. In $\mathsf{dg}(\Sigma)$ there is a cycle that contains a special edge. However, there exists only one so-chase derivation of $\mathsf{cr}(\Sigma)$ w.r.t. Σ that is finite, and thus, $\Sigma \in \mathbb{CT}^{\mathsf{so}}_{\forall}$.

Interestingly, as it has been shown in [8], there is an extension of weak-acyclicity, called critical-weak-acyclicity, that, whenever we focus on linear TGDs, provides a necessary and sufficient condition for the termination of the semi-oblivious chase. A key notion underlying critical-weak-acyclicity is the notion of compatibility among two single-head linear TGDs. Intuitively, if a single-head linear TGD σ_1 is compatible with a single-head linear TGD σ_2, then the atom obtained during the chase by applying σ_1 may trigger σ_2. To formally define the notion of compatibility, we first need to recall the standard notion of unification among atoms.

We say that two atoms α and β (containing only variables from $\mathbf{V}$) *unify* if there exists a substitution γ from the variables occurring in α and β to $\mathbf{V}$, called *unifier* for α and β, such that $\gamma(\alpha) = \gamma(\beta)$. A *most general unifier* (MGU) for α and β is a unifier for α and β, denoted $\mathsf{mgu}(\alpha, \beta)$, such that, for each other unifier γ for α and β, there exists a substitution γ' such that $\gamma = \gamma' \circ \mathsf{mgu}(\alpha, \beta)$. It is well-known that if two atoms α and β unify, then they have an MGU that is unique (modulo variable renaming), and thus, we can refer to *the* MGU for α and β [1]. For brevity, given a TGD σ and a variable x, let $\Pi^{\sigma}_{x} = \mathsf{pos}(\mathsf{body}(\sigma), x)$. Let $\mathsf{var}(\alpha, \Pi)$, where α is an atom, and Π a set of positions, be the set of variables in α at positions of Π.

Definition 5 (**Compatibility**) Consider two single-head linear TGDs σ_1 and σ_2. We say that σ_1 is *compatible* with σ_2 if the following hold:

1. $\mathsf{head}(\sigma_1)$ and $\mathsf{body}(\sigma_2)$ unify.
2. For each $x \in \mathsf{var}(\mathsf{body}(\sigma_2))$, either $\mathsf{var}(\mathsf{head}(\sigma_1), \Pi^{\sigma_2}_{x}) \subseteq \mathsf{fr}(\sigma_1)$, or there is an existentially quantified variable z in σ_1 such that $\mathsf{var}(\mathsf{head}(\sigma_1), \Pi^{\sigma_2}_{x}) = \{z\}$.

Having the notion of compatibility among two single-head linear TGDs in place, we can recall the notion of resolvent of a sequence $\sigma_1, \ldots, \sigma_n$ of single-head linear TGDs, which is in turn a single-head TGD. Roughly, such a resolvent mimics the behavior of the sequence $\sigma_1, \ldots, \sigma_n$ during the chase. Notice that the existence of such a resolvent is not guaranteed, but if it exists, this implies that we may have a sequence of trigger applications that involve the TGDs $\sigma_1, \ldots, \sigma_n$ in this order. In such a case, we call the sequence $\sigma_1, \ldots, \sigma_n$ active.

Definition 6 (**Resolvent**) The *resolvent* of a sequence $\sigma_1, \ldots, \sigma_n$ of single-head linear TGDs, denoted $[\sigma_1, \ldots, \sigma_n]$, is inductively defined as follows (for notational convenience, we simply write ρ for $[\sigma_1, \ldots, \sigma_{n-1}]$):

1. $[\sigma_1] = \sigma_1$.

2. $[\sigma_1, \ldots, \sigma_n] = \gamma(\mathsf{body}(\rho)) \rightarrow \gamma(\mathsf{head}(\sigma_n))$, with $\gamma = \mathsf{mgu}(\mathsf{head}(\rho), \mathsf{body}(\sigma_n))$, if $\rho \neq \blacklozenge$ and ρ is compatible with σ_n; otherwise, $[\sigma_1, \ldots, \sigma_n] = \blacklozenge$.

The sequence $\sigma_1, \ldots, \sigma_n$ is called *active* if $[\sigma_1, \ldots, \sigma_n] \neq \blacklozenge$.

At this point, one may think that the right extension of weak-acyclicity, which will provide a necessary condition for the termination of the semi-oblivious chase under linear TGDs, is to allow cycles with special edges in the underlying dependency graph as long as the corresponding sequence of single-head TGDs, which can be extracted from the edge labels, is not active. However, as thoroughly discussed in [8], this is not enough. If a cycle with a special edge is labeled with an active sequence, then we can conclude that it will be traversed at least once during the chase. Nevertheless, it is not guaranteed that it will be traversed infinitely many times. A cycle that is labeled with an active sequence $\sigma_1, \ldots, \sigma_n$, and contains a special edge, will be certainly traversed infinitely many times if the resolvent of the sequence $\rho, \ldots, \rho$ of length k, where $\rho = [\sigma_1, \ldots, \sigma_n]$, exists, for every $k > 0$. Interestingly, for ensuring the latter condition, it suffices to consider sequences of length at most $(\omega + 1)$, where ω is the arity of the predicate of $\mathsf{body}(\sigma_1)$. This brings us to critical sequences. For brevity, we write σ^k for the sequence $\sigma, \ldots, \sigma$ of length k.

Definition 7 (**Critical Sequence**) A sequence $\sigma_1, \ldots, \sigma_n$ of single-head linear TGDs is *critical* if $\sigma_1, \ldots, \sigma_n$ is active, and $[\sigma_1, \ldots, \sigma_n]^{\omega+1}$ is active, where ω is the arity of the predicate of $\mathsf{body}(\sigma_1)$.

We can now recall critical-weak-acyclicity as defined in [8]. It is essentially weak-acyclicity, with the key difference that a cycle in the underlying graph is "dangerous", not only if it contains a special edge, but if it is also labeled with a critical sequence of single-head linear TGDs. The formal definition follows.

Definition 8 (**Critical-Weak-Acyclicity**) Consider a set $\Sigma \in \mathbb{L}$ of TGDs, and let $\mathsf{dg}(\Sigma) = (N, E, \lambda)$. A cycle $v_0, v_1, \ldots, v_n, v_0$ in $\Sigma g \Sigma$ is *critical* if the sequence $\lambda(v_0, v_1), \lambda(v_1, v_2), \ldots, \lambda(v_n, v_0)$ of single-head linear TGDs is critical. We call Σ *critically-weakly-acyclic* if no critical cycle in $\mathsf{dg}(\Sigma)$ contains a special edge.

The essence of critical-weak-acyclicity is revealed by the following result:

Theorem 4 ([8]) *Consider a set $\Sigma \in \mathbb{L}$ of TGDs. The following are equivalent:*

1. $\Sigma \in \mathbb{CT}^{\mathsf{so}}_{\forall}$.
2. *Σ is critically-weakly-acyclic.*

5.2 From Sticky to Linear TGDs

Before presenting the linearization procedure, we need to introduce some auxiliary notions. Given a tuple $\bar{t} = (t_1, \ldots, t_n) \in (\mathbf{V} \cup \{\natural\})^n$, we write $\mathsf{shape}(\bar{t})$ for the

tuple obtained from $\bar{t}$ by replacing each variable of $\mathbf{V}$ with the special symbol $*$. For example, $\mathsf{shape}((x, y, \natural, z, x, \natural)) = (*, *, \natural, *, *, \natural)$. We also write $\mathsf{svar}(\bar{t})$ for the tuple obtained from $\bar{t}$ by removing all the occurrences of the constant $\natural$. For example, $\mathsf{svar}((x, y, \natural, z, x, \natural)) = (x, y, z, x)$. For an atom $\alpha = R(\bar{t})$, let

$$\natural\text{-}\mathsf{free}(\alpha) \;=\; R_{\mathsf{shape}(\bar{t})}(\mathsf{svar}(\bar{t})),$$

where $R_{\mathsf{shape}(\bar{t})}$ is of arity $|\mathsf{svar}(\bar{t})|$. In fact, $\natural\text{-}\mathsf{free}(\alpha)$ is the constant-free version of α, while the subscript $\mathsf{shape}(\bar{t})$ keeps track of the original shape of α, i.e., where each occurrence of $\natural$ occurs in α. Notice that, having the constant-free version of an atom α in place, we can unambiguously write down α. For a set of atoms A, let

$$\natural\text{-}\mathsf{free}(A) \;=\; \{\natural\text{-}\mathsf{free}(\alpha) \mid \alpha \in A\}.$$

Given a TGD σ and an atom $\alpha \in \mathsf{body}(\sigma)$, let

$$V_{\alpha,\sigma} \;=\; \mathsf{var}(\mathsf{body}(\sigma) \setminus \{\alpha\}),$$

that is, the set of body variables of σ that do not occur only in α. Now, given a TGD σ, and an atom $\alpha \in \mathsf{body}(\sigma)$, let

$$M_{\alpha,\sigma} \;=\; \{h : T \to \{\natural\} \mid V_{\alpha,\sigma} \subseteq T \subseteq \mathsf{var}(\mathsf{body}(\sigma))\},$$

i.e., for each subset T of $\mathsf{var}(\mathsf{body}(\sigma))$ that contains all the variables of $V_{\alpha,\sigma}$, $M_{\alpha,\sigma}$ contains a mapping that maps each variable of T to the special constant $\natural$.

We are now ready to introduce the linearization of a set of TGDs. Note that the following definition talks about arbitrary TGDs. The notion of stickiness is used later for showing that the termination of the semi-oblivious chase is preserved.

Definition 9 (Linearization) The *linearization* of a TGD σ of the form $\phi(\bar{x}, \bar{y}) \to \exists \bar{z}\, \psi(\bar{x}, \bar{z})$, denoted $\mathsf{Lin}(\sigma)$, is the set of linear TGDs

$$\bigcup_{\alpha \in \phi(\bar{x},\bar{y})} \; \bigcup_{h \in M_{\alpha,\sigma}} \{\natural\text{-}\mathsf{free}(h(\alpha)) \to \exists \bar{z}\, \natural\text{-}\mathsf{free}(h(\psi(\bar{x}, \bar{z})))\}\,.$$

The linearization of a set Σ of TGDs is defined as $\mathsf{Lin}(\Sigma) = \bigcup_{\sigma \in \Sigma} \mathsf{Lin}(\sigma)$.

The linearization procedure converts a TGD σ into a set of (constant-free) linear TGDs, where the body atom of each such linear TGD corresponds to an atom α of $\mathsf{body}(\sigma)$, while the variables in $\mathsf{body}(\sigma) \setminus \{\alpha\}$, and possibly additional variables that occur only in α, are instantiated with the special constant $\natural$. An example follows:

Example 10 Consider the TGD

$$\sigma \;=\; \underbrace{P(x, y, z)}_{\alpha_1}, \underbrace{R(y, w)}_{\alpha_2} \to \exists u S(w, y, u),$$

We have that the set $V_{\alpha_1,\sigma}$ consists of all the variables in $\mathsf{body}(\sigma) \setminus \{\alpha_1\}$, while $V_{\alpha_2,\sigma}$ of all the variables in $\mathsf{body}(\sigma) \setminus \{\alpha_2\}$, that is,

$$V_{\alpha_1,\sigma} = \{y, w\} \qquad V_{\alpha_2,\sigma} = \{x, y, z\}.$$

There are four sets of variables in $\mathsf{body}(\sigma)$ that are supersets of $V_{\alpha_1,\sigma}$:

$$T_1 = \{y, w\} \qquad T_2 = \{x, y, w\} \qquad T_3 = \{y, z, w\} \qquad T_4 = \{x, y, z, w\}.$$

Moreover, there are two sets of variables in $\mathsf{body}(\sigma)$ that are supersets of $V_{\alpha_2,\sigma}$:

$$T'_1 = \{x, y, z\} \qquad T'_2 = \{x, y, z, w\}.$$

Therefore, the set of mappings $M_{\alpha_1,\sigma}$ contains, for each $i \in \{1, \ldots, 4\}$, a mapping of the form $\{x \mapsto \natural \mid x \in T_i\}$, whereas $M_{\alpha_2,\sigma}$ contains, for each $i \in \{1, 2\}$, a mapping of the form $\{x \mapsto \natural \mid x \in T'_i\}$. Finally, the linearization of σ is the set of linear TGDs $\mathsf{Lin}(\sigma)$ consisting of the TGDs

$$\begin{aligned} P_{(*,\natural,*)}(x, z) &\rightarrow \exists u\, S_{(\natural,\natural,*)}(u), \\ P_{(\natural,\natural,*)}(z) &\rightarrow \exists u\, S_{(\natural,\natural,*)}(u), \\ P_{(*,\natural,\natural)}(x) &\rightarrow \exists u\, S_{(\natural,\natural,*)}(u), \\ P_{(\natural,\natural,\natural)}() &\rightarrow \exists u\, S_{(\natural,\natural,*)}(u), \end{aligned}$$

due to the set of mappings $M_{\alpha_1,\sigma}$, and the TGDs

$$\begin{aligned} R_{(\natural,*)}(w) &\rightarrow \exists u\, S_{(*,\natural,*)}(w, u), \\ R_{(\natural,\natural)}() &\rightarrow \exists u\, S_{(\natural,\natural,*)}(u). \end{aligned}$$

due to the set of mappings $M_{\alpha_2,\sigma}$.

Lemma 3 allows us to show that this procedure preserves the termination of the semi-oblivious chase whenever the input set of TGDs is sticky.

Lemma 3 *For every set $\Sigma \in \mathbb{S}$ of TGDs, $\Sigma \in \mathbb{CT}^{\mathsf{so}}_{\forall}$ iff $\mathsf{Lin}(\Sigma) \in \mathbb{CT}^{\mathsf{so}}_{\forall}$.*

Proof We assume, w.l.o.g., that Σ is in normal form, i.e., each TGD of Σ has only one atom in its head. Given a TGD $\sigma = \phi(\bar{x}, \bar{y}) \rightarrow \exists \bar{z}\ R(\bar{x}, \bar{z})$ of Σ, an atom $\alpha \in \mathsf{body}(\sigma)$, and a mapping $h \in M_{\alpha,\sigma}$, we write $\sigma^{\alpha,h}$ for the linear TGD $\natural\text{-}\mathsf{free}(h(\alpha)) \rightarrow \exists \bar{z}\ \natural\text{-}\mathsf{free}(h(R(\bar{x}, \bar{z}))) \in \mathsf{Lin}(\Sigma)$. We are now ready to proceed with the proof.

($\Rightarrow$) Assume that $\mathsf{Lin}(\Sigma) \notin \mathbb{CT}^{\mathsf{so}}_{\forall}$. By Corollary 1, there exists an infinite so-chase derivation $\delta = (I_i)_{i \geq 0}$ of $\mathsf{cr}(\mathsf{Lin}(\Sigma))$ w.r.t. $\mathsf{Lin}(\Sigma)$, with $I_i \langle \sigma_i^{\alpha_i, g_i}, h_i \rangle I_{i+1}$, or, equivalently, $I_{i+1} = I_i \cup \mathsf{so\text{-}result}(\sigma_i^{\alpha_i, g_i}, h_i)$, for each $i \geq 0$. By construction, for

each $i \geq 0$, the TGD $\sigma_i^{\alpha_i, g_i}$ is linear, with its body being an atom of the form $R_{\bar{t}}(\bar{x})$, where $\bar{x}$ is a tuple of variables, and $\bar{t}$ is a tuple over $\{*, \natural\}$. Let $\hat{h}_i$ be the extension of h_i that maps each variable in $\mathsf{body}(\sigma_i)$ that is not in $\bar{x}$ to $\natural$.

Consider the infinite sequence of instances $\delta' = (J_i)_{i \geq 0}$, where $J_0 = \mathsf{cr}(\Sigma)$, and $J_{i+1} = J_i \cup \mathsf{so\text{-}result}(\sigma_i, \hat{h}_i)$, for each $i \geq 0$. By construction of $\hat{h}_i$, for each $i \geq 0$, $(\sigma_i, \hat{h}_i)$ is a trigger for Σ on J_i, which implies that $J_i \langle \mathsf{so}, \sigma_i, \hat{h}_i \rangle J_{i+1}$. However, the above sequence is not necessarily an infinite so-chase derivation of Σ w.r.t. $\mathsf{cr}(\Sigma)$ since some of the involved triggers may not be semi-obliviously active. We therefore consider the infinite sequence of instances $\delta'' = (J'_i)_{i \geq 0}$ obtained from $(J_i)_{i \geq 0}$ by simply removing the triggers that are not semi-obliviously active, or, more formally, by removing the instances obtained from triggers that are not semi-obliviously active. Recall that, for each $i > 0$, J_i is obtained via a trigger that is not semi-obliviously active iff $J_i = J_{i-1}$. It remains to show that δ'' is infinite, which in turn implies that $\Sigma \notin \mathbb{CT}^{\mathsf{so}}_{\forall}$. It suffices to show that, for each $i \geq 0$, J_i occurs in δ' finitely many times.

By contradiction, assume that there is $k \geq 0$ such that J_k occurs in δ' infinitely many times, which means that $J_k = J_{k+1} = J_{k+2} = \cdots$. Since $\mathsf{Lin}(\Sigma)$ is finite, there are indices $k \leq j < i$ such that $\sigma_i^{\alpha_i, g_i} = \sigma_j^{\alpha_j, g_j}$; we refer to those TGDs as ρ. Since $(I_i)_{i \geq 0}$ is a so-chase derivation of $\mathsf{Lin}(\Sigma)$ w.r.t. $\mathsf{cr}(\mathsf{Lin}(\Sigma))$, there is a variable $x \in \mathsf{fr}(\rho)$ such that $h_i(x) \neq h_j(x)$. By definition, $h_i(x) = \hat{h}_i(x)$ and $h_j(x) = \hat{h}_j(x)$. This implies that $\hat{h}_i(x) \neq \hat{h}_j(x)$, and therefore, $J_i \neq J_j$, which is a contradiction.

($\Leftarrow$) Assume now that $\Sigma \notin \mathbb{CT}^{\mathsf{so}}_{\forall}$. By Theorem 3, there exists a linear infinite so-chase derivation $\delta = (I_i)_{i \geq 0}$ of $\mathsf{cr}(\Sigma)$ w.r.t. Σ, with $I_i \langle \sigma_i, h_i \rangle I_{i+1}$, for $i \geq 0$. In other words, there exists an infinite sequence of distinct atoms $(\alpha_i)_{i \geq 0}$ such that

1. $\alpha_0 \in \mathsf{cr}(\Sigma)$, and
2. for each $i \geq 0$, $\alpha_{i+1} \in I_{i+1} \setminus I_i$, and there exists an atom $\beta_i \in \mathsf{body}(\sigma_i)$ such that $\alpha_i = h_i(\beta_i)$ and $h_i(\mathsf{body}(\sigma_i) \setminus \{\beta_i\}) \subseteq \mathsf{cr}(\Sigma)$.

For every $i \geq 0$, let $X_i = \mathsf{var}(\mathsf{body}(\sigma_i) \setminus \{\beta_i\})$, $g_i = h_{i|X_i}$, and $f_i = h_i \setminus g_i$. Note that, since δ is linear, for every $x \in X_i$, $g_i(x) = \natural$. Assuming that σ_i is of the form $\phi_i(\bar{x}_i, \bar{y}_i) \rightarrow \exists \bar{z}_i \; R_i(\bar{x}_i, \bar{z}_i)$, let

$$\Sigma' = \left\{ \underbrace{\natural\text{-}\mathsf{free}(g_i(\beta_i)) \rightarrow \exists \bar{z}_i \natural\text{-}\mathsf{free}(g_i(R_i(\bar{x}_i, \bar{z}_i)))}_{\sigma'_i} \right\}_{i \geq 0} .$$

By construction, $\Sigma' \subseteq \mathsf{Lin}(\Sigma)$, and therefore, $\Sigma' \notin \mathbb{CT}^{\mathsf{so}}_{\forall}$ implies $\mathsf{Lin}(\Sigma) \notin \mathbb{CT}^{\mathsf{so}}_{\forall}$. Hence, to conclude the proof of Lemma 3, it suffices to exhibit an infinite so-chase derivation of $\mathsf{cr}(\Sigma') \subseteq \mathsf{cr}(\Sigma)$ w.r.t. Σ'. To this end, consider the infinite sequence of instances $\delta' = (J_i)_{i \geq 0}$, where $J_0 = \mathsf{cr}(\Sigma')$, and $J_i = J_{i-1} \cup \{\alpha'_i\}$, where $\alpha'_i = f_i(\mathsf{body}(\sigma'_i)) = f_i(\natural\text{-}\mathsf{free}(g_i(\beta_i)))$, for each $i > 0$. It is not difficult to verify that δ' is an infinite so-chase derivation of $\mathsf{cr}(\Sigma')$ w.r.t. Σ' (modulo null renaming). □

The main result of this section follows from Theorem 4 and Lemma 3:

Theorem 5 *Consider a set $\Sigma \in \mathbb{S}$ of TGDs. The following are equivalent:*

1. $\Sigma \in \mathbb{CT}^{\mathsf{so}}_{\forall}$.
2. $\mathsf{Lin}(\Sigma)$ *is critically-weakly-acyclic.*

We would like to remark that the above characterization of the termination of the so-chase is different than the one presented in the conference paper [10] (see Corollary 24). More precisely, the linearization procedure in [10] produces TGDs *with* constants, and thus, we had to properly extend the notion of critical-weak-acyclicity from [8], which was defined only for constant-free TGDs. It turned out that we can directly use the notion of critical-weak-acyclicity as defined in [8], at the price of a slightly more complex linearization procedure that produces constant-free TGDs.

6 Complexity Analysis

We are now ready to complete the proof of our main result, that is, Theorem 2, which states that $\mathsf{CT}^{\mathsf{o}}_{\forall}(\mathbb{S})$ and $\mathsf{CT}^{\mathsf{so}}_{\forall}(\mathbb{S})$ are PSPACE-complete, and NLOGSPACE-complete for predicates of bounded arity. We first concentrate on the upper bounds.

6.1 Upper Bounds for $\mathsf{CT}^{\mathsf{so}}_{\forall}(\mathbb{S})$

We know that the problem $\mathsf{CT}^{\mathsf{so}}_{\forall}(\mathbb{L})$ is in PSPACE, and in NLOGSPACE for predicates of bounded arity [8]. In fact, these results exploit Theorem 4, and are obtained by showing that deciding whether a set of linear TGDs is critically-weakly-acyclic is in PSPACE, and in NLOGSPACE for predicates of bounded arity. However, despite the fact that we can reduce $\mathsf{CT}^{\mathsf{so}}_{\forall}(\mathbb{S})$ to $\mathsf{CT}^{\mathsf{so}}_{\forall}(\mathbb{L})$ (see Lemma 3), we cannot directly exploit the complexity results for linear TGDs. The reason is because the linearization procedure takes exponential time, in general, and polynomial time in the case of bounded-arity predicates. Therefore, we cannot simply compute the set $\mathsf{Lin}(\Sigma)$, and then check for critical-weak-acyclicity, but a more refined approach is needed.

We focus on the complement of our problem, i.e., given a set $\Sigma \in \mathbb{S}$ of TGDs, we want to check whether $\Sigma \notin \mathbb{CT}^{\mathsf{so}}_{\forall}$. By Theorem 5, it suffices to show that $\mathsf{Lin}(\Sigma)$ is not critically-weakly-acyclic. The latter problem can be seen as a generalization of the standard graph reachability problem. Indeed, we need to check whether there exists a node in the dependency graph of $\mathsf{Lin}(\Sigma)$ that is reachable from itself via a critical cycle that contains a special edge. However, as discussed above, we cannot explicitly construct $\mathsf{Lin}(\Sigma)$ and its dependency graph G. Instead, the above reachability check should be performed on a compact representation of G, which is the set Σ itself.

Lemma 4 *Consider a set Σ of TGDs. The problem of deciding whether $\mathsf{Lin}(\Sigma)$ is not critically-weakly-acyclic is in*

$$\mathrm{NSPACE}(\log|\Sigma| + \omega \cdot \log(\omega \cdot |\mathsf{sch}(\Sigma)| \cdot 2^{\omega}) + \omega \cdot \log m),$$

where $\omega = \mathsf{ar}(\Sigma)$, and m is the number of variables occurring in Σ.

Algorithm 1: Check that $\mathsf{Lin}(\Sigma)$ is not critically-weakly-acyclic.

Input: A set Σ of TGDs
Output: Accept if $\mathsf{Lin}(\Sigma)$ is not critically-weakly-acyclic; otherwise, Reject

guess a Σ-label $\ell = (\sigma, \alpha, \beta, T)$ and positions $v_1, v_2 \in \mathsf{pos}(\mathsf{sch}(\mathsf{Lin}(\Sigma)))$
$(edge, special) := \mathsf{isEdge}(\ell, v_1, v_2)$
if $edge = 0$ **then**
  Reject
if $special = 1$ **then**
  $flag := 1$
$\rho := \tau(\ell)$
repeat
  guess a Σ-label $\ell = (\sigma, \alpha, \beta, T)$ and a position $v_3 \in \mathsf{pos}(\mathsf{sch}(\mathsf{Lin}(\Sigma)))$
  $(edge, special) := \mathsf{isEdge}(\ell, v_2, v_3)$
  if $edge = 0$ **then**
    Reject
  if $special = 1$ **then**
    $flag := 1$
  if ρ is not compatible with $\tau(\ell)$ **then**
    Reject
  else
    $v_2 := v_3$ and $\rho := [\rho, \tau(\ell)]$
until $v_1 = v_3$;
if $flag = 0$ **then**
  Reject
if $[\rho]^{\omega+1}$ *is active* **then**
  Accept
else
  Reject

Algorithm 2: Check for the existence of a normal or special edge.

Input: A Σ-label $\ell = (\sigma, \alpha, \beta, T)$ and positions
$v = (R, i), u = (P, j) \in \mathsf{pos}(\mathsf{sch}(\mathsf{Lin}(\Sigma)))$
Output: $(1, 0)$ (resp., $(1, 1)$, $(0, 0)$) if (v, u) is a normal (resp., special, neither normal nor special) edge of $dg(\mathsf{Lin}(\Sigma))$ with label $\tau(\ell)$
if R is not the predicate of $\mathsf{body}(\tau(\ell))$ *or* P is not the predicate of $\mathsf{head}(\tau(\ell))$ **then**
  return $(0, 0)$
if $\mathsf{body}(\tau(\ell))[i] \neq \mathsf{head}(\tau(\ell))[j]$ **then**
  if $\mathsf{body}(\tau(\ell))[i] \notin \mathsf{fr}(\tau(\ell))$ *or* $\mathsf{head}(\tau(\ell))[j] \in \mathsf{fr}(\tau(\ell))$ **then**
    return $(0, 0)$
  else
    return $(1, 1)$
else
  return $(1, 0)$

Proof We employ Algorithm 1, which takes as input a set Σ of TGDs, and checks whether $\mathsf{Lin}(\Sigma)$ is not critically-weakly-acyclic by non-deterministically searching for the existence of a critical cycle in $\mathsf{dg}(\mathsf{Lin}(\Sigma))$ that contains a special edge. Note that this is done without explicitly computing the graph $\mathsf{dg}(\mathsf{Lin}(\Sigma))$. Before showing that Algorithm 1 is correct, and analyzing its space complexity, let us first introduce an auxiliary notion that is used by the algorithm.

A *Σ-label* is a tuple $\ell = (\sigma, \alpha, \beta, T)$, where $\sigma \in \Sigma$, $\alpha \in \mathsf{body}(\sigma)$, $\beta \in \mathsf{head}(\sigma)$, and $V_{\alpha,\sigma} \cap (\mathsf{var}(\alpha) \cup \mathsf{var}(\beta)) \subseteq T \subseteq \mathsf{var}(\mathsf{body}(\sigma)) \cap (\mathsf{var}(\alpha) \cup \mathsf{var}(\beta))$. Note that a Σ-label induces a single-head TGD of $\mathsf{Lin}(\Sigma)$, denoted $\tau(\ell)$, which might be the label of an edge in $\mathsf{dg}(\mathsf{Lin}(\Sigma))$; hence the name Σ-label. In fact, $\tau(\ell)$ is the TGD

$$\natural\text{-}\mathsf{free}(h(\alpha)) \rightarrow \exists \bar{z} \natural\text{-}\mathsf{free}(h(\beta)),$$

where h maps each variable of T to $\natural$, and $\bar{z}$ are the existentially quantified variables of σ occurring in β. Let us now proceed with the correctness of Algorithm 1.

Correctness It is an easy exercise to show the following claim concerning the procedure $\mathsf{isEdge}(\cdot, \cdot, \cdot)$, which is described in Algorithm 2. Let $\mathsf{dg}(\mathsf{Lin}(\Sigma)) = (N, E, \lambda)$.

Claim Consider a Σ-label ℓ, and two positions $v, u \in \mathsf{sch}(\mathsf{Lin}(\Sigma))$. Then:

1. $\mathsf{isEdge}(\ell, v, u) = (1, \_)$ iff $(v, u) \in E$ and $\lambda((v, u)) = \tau(\ell)$.
2. $\mathsf{isEdge}(\ell, v, u) = (1, 1)$ iff $(v, u) \in E$ is a special edge.

We can now show that Algorithm 1 accepts on input Σ iff there is a critical cycle in $\mathsf{dg}(\mathsf{Lin}(\Sigma))$ that contains a special edge:

($\Rightarrow$) Assume first that Algorithm 1 accepts. This implies that there is an accepting computation that guesses a sequence of Σ-labels $\ell_1, \ldots, \ell_n$, and a sequence of positions $v_0, \ldots, v_n$ of $\mathsf{pos}(\mathsf{sch}(\mathsf{Lin}(\Sigma)))$. By construction, the following hold:

- For each $i \in [n]$, $\mathsf{isEdge}(\ell_i, v_{i-1}, v_i) = (1, \_)$, which implies that $(v_{i-1}, v_i) \in E$ and $\lambda((v_{i-1}, v_i)) = \tau(\ell)$.
- There exists $i \in [n]$ such that $\mathsf{isEdge}(\ell_i, v_{i-1}, v_i) = (1, 1)$. Hence, $(v_{i-1}, v_i) \in E$ is a special edge.
- For each $i \in [n-1]$, $[\tau(\ell_1), \ldots, \tau(\ell_{i-1})]$ is compatible with $\tau(\ell_{i+1})$, which in turn implies that $\rho = [\tau(\ell_1), \ldots, \tau(\ell_n)]$ is active.
- $v_0 = v_n$.
- $[\rho]^{\omega+1}$ is active.

The above properties imply that $v_0, \ldots, v_{n-1}, v_0$ is a critical cycle in $\mathsf{dg}(\mathsf{Lin}(\Sigma))$ that contains at least one special edge, and the claim follows.

($\Leftarrow$) By hypothesis, there exists a critical cycle $v_0, \ldots, v_{n-1}, v_0$ in $\mathsf{dg}(\mathsf{Lin}(\Sigma))$. Assume that, for each $i \in [n-1]$, $\lambda((v_{i-1}, v_i)) = \sigma_i$ and $\lambda((v_{n-1}, v_0)) = \sigma_n$. Observe that each single-head linear TGD σ_i, for $i \in [n]$, corresponds to a Σ-label ℓ_i. Consider now the computation of Algorithm 1 on input Σ that guesses the sequence of labels $\ell_1, \ldots, \ell_n$, and the sequence of positions $v_0, \ldots, v_{n-1}, v_0$. It should be clear that this is an accepting computation, and the claim follows.

Space Complexity It remains to show that the space needed at each step of a computation of Algorithm 1 on input Σ is

$$O(\log|\Sigma| + \omega \cdot \log(\omega \cdot |\mathsf{sch}(\Sigma)| \cdot 2^{\omega}) + \omega \cdot \log m).$$

We proceed to analyze the space needed to store a Σ-label, a position of $\mathsf{sch}(\mathsf{Lin}(\Sigma))$, and a single-head linear TGD $\tau(\ell)$ for a Σ-label ℓ. We also analyze the space for a compatibility check, for constructing a resolvent, and for checking whether $[\rho]^{\omega+1}$ is active. It will be then apparent that indeed the space used at each step of a computation of Algorithm 1 on input Σ is what we claimed above. Note that the latter ensures the termination of Algorithm 1 since we can always force a space-bounded algorithm to terminate and reject after exponentially many steps in the required space [31].

- The required space for a Σ-label $\ell = (\sigma, \alpha, \beta, T)$ is

  $$O(\log|\Sigma| + \log|\mathsf{sch}(\Sigma)| + \omega \cdot \log m).$$

  A TGD $\sigma \in \Sigma$ takes $O(\log|\Sigma|)$ space. For storing an atom occurring in Σ we need to store a predicate of $\mathsf{sch}(\Sigma)$, and, in the worst-case, ω variables occurring in Σ, which can be done in space $O(\log|\mathsf{sch}(\Sigma)| + \omega \cdot \log m)$. Finally, since T contains at most 2ω variables occurring in Σ, it can be stored in space $O(\omega \cdot \log m)$. Summing up, ℓ requires the space stated above.
- A position of $\mathsf{sch}(\mathsf{Lin}(\Sigma))$ takes space

  $$O(\log(\omega \cdot |\mathsf{sch}(\Sigma)| \cdot 2^{\omega})).$$

  This follows from the fact that, by construction, the number of predicates occurring in $\mathsf{Lin}(\Sigma)$ is at most $|\mathsf{sch}(\Sigma)| \cdot 2^{\omega}$.
- Given a Σ-label ℓ, the single-head linear TGD $\tau(\ell)$ takes space

  $$O(\log(|\mathsf{sch}(\Sigma)| \cdot 2^{\omega}) + \omega \cdot \log m).$$

  It is clear that the two predicates occurring in $\tau(\ell)$ require $O(|\mathsf{sch}(\Sigma)| \cdot 2^{\omega})$ space. We also need to store, in the worst-case, 2ω variables occurring in Σ, which takes $O(\omega \cdot \log m)$ space. Summing up, $\tau(\ell)$ requires the space stated above.
- To check whether two single-head linear TGDs σ_1, σ_2, computed during the execution of the algorithm, are compatible, we only need to check that, for each $x \in \mathsf{var}(\mathsf{body}(\sigma_2))$, either $\mathsf{var}(\mathsf{head}(\sigma_1), \Pi_x^{\sigma_2}) \subseteq \mathsf{fr}(\sigma_1)$, or there is an existentially quantified variable z in σ_1 such that $\mathsf{var}(\mathsf{head}(\sigma_1), \Pi_x^{\sigma_2}) = \{z\}$. The latter can be performed using space

  $$O(\omega \cdot \log(\omega \cdot |\mathsf{sch}(\Sigma)| \cdot 2^{\omega})),$$

  which is the space needed for storing $\Pi_x^{\sigma_2}$. Note that $\mathsf{head}(\sigma_1)$ and $\mathsf{body}(\sigma_2)$ always unify since, at the point that we perform the compatibility check, we know that they have the same predicate (this has been checked by isEdge), and thus, there is no way for the unification check to fail.
- Given two single-head linear TGDs σ_1, σ_2, computed during the execution of the algorithm, that are compatible, constructing $[\sigma_1, \sigma_2]$ can be done using space

  $$O(\omega \cdot \log m),$$

which is essentially the space needed for storing $\mathsf{mgu}(\mathsf{head}(\sigma_1), \mathsf{body}(\sigma_2))$.[10]

– Finally, given the single-head linear TGD ρ, computed after the execution of the repeat-until, checking whether $[\rho]^{\omega+1}$ is active can be done using space

$$O(\omega \cdot \log(\omega \cdot |\mathsf{sch}(\Sigma)| \cdot 2^{\omega}) + \omega \cdot \log m).$$

This easily follows from the analysis performed above.

Summing up, each step of a computation of Algorithm 1 on input Σ takes space

$$O(\log |\Sigma| + \omega \cdot \log(\omega \cdot |\mathsf{sch}(\Sigma)| \cdot 2^{\omega}) + \omega \cdot \log m),$$

and the claim follows. □

Having Lemma 4 in place, it is clear that the complement of $\mathsf{CT}^{\mathsf{so}}_{\forall}(\mathbb{S})$, and thus $\mathsf{CT}^{\mathsf{so}}_{\forall}(\mathbb{S})$ itself, is in PSPACE, and in NLOGSPACE for predicates of bounded arity.[11]

6.2 Upper Bounds for $\mathsf{CT}^{\mathsf{o}}_{\forall}(\mathbb{S})$

We proceed to explain how the upper bounds established above for $\mathsf{CT}^{\mathsf{so}}_{\forall}(\mathbb{S})$ can be transferred to $\mathsf{CT}^{\mathsf{o}}_{\forall}(\mathbb{S})$. Since the semi-oblivious chase is a refined version of the oblivious chase, it is not surprising that $\mathsf{CT}^{\mathsf{o}}_{\forall}(\mathbb{TGD})$ can be reduced to $\mathsf{CT}^{\mathsf{so}}_{\forall}(\mathbb{TGD})$. This relies on a very simple construction, known as enrichment [20]. Formally, the *enrichment* of a set Σ of TGDs, denoted $\mathsf{enrichment}(\Sigma)$, is the set of TGDs obtained by replacing each TGD $\sigma \in \Sigma$ of the form $\phi(\bar{x}, \bar{y}) \rightarrow \exists \bar{z}\ \psi(\bar{x}, \bar{z})$ with the TGD

$$\phi(\bar{x}, \bar{y}) \rightarrow \exists \bar{z}\ \psi(\bar{x}, \bar{z}), \mathrm{Aux}_{\sigma}(\bar{x}, \bar{y}),$$

where Aux_{σ} is an auxiliary predicate of arity $(|\bar{x}| + |\bar{y}|)$ not occurring in $\mathsf{sch}(\Sigma)$. It is easy to show that the notion of enrichment provides the desired reduction from $\mathsf{CT}^{\mathsf{o}}_{\forall}(\mathbb{TGD})$ to $\mathsf{CT}^{\mathsf{so}}_{\forall}(\mathbb{TGD})$. More precisely:

Lemma 5 *For a set Σ of TGDs, the following are equivalent:*

1. $\Sigma \in \mathbb{CT}^{\mathsf{o}}_{\forall}$.
2. $\mathsf{enrichment}(\Sigma) \in \mathbb{CT}^{\mathsf{so}}_{\forall}$.

Proof The fact that (2) implies (1) is a consequence of Theorem 6.1 in [20]. We proceed to show that (1) implies (2). Assume that $\mathsf{enrichment}(\Sigma) \notin \mathbb{CT}^{\mathsf{so}}_{\forall}$. Therefore, there exists a database D, and an infinite so-chase derivation $\delta = (I_i)_{i \geq 0}$ of D w.r.t. $\mathsf{enrichment}(\Sigma)$, where $I_i \langle \mathsf{so}, \sigma_i', h_i \rangle$, I_{i+1}, and, for each $i \geq 0$, $\sigma_i' \in \mathsf{enrichment}(\Sigma)$ is the TGD corresponding to some TGD $\sigma_i \in \Sigma$. We can easily construct an infinite o-chase derivation $\delta' = (J_i)_{i \geq 0}$ of D w.r.t. Σ: let $J_0 = D$, and, for each $i > 0$, J_i is obtained from I_i by removing every atom with a predicate symbol of the form Aux_{σ}, for some $\sigma \in \Sigma$, and by replacing every null of the form $\perp^{z}_{\sigma', h}$ with the null

[10] For constructing the MGU, one can use a simplified version of Robinson's unification algorithm that does not consider functions symbols.

[11] Recall that by the Immerman-Szelepcsényi Theorem, coNLOGSPACE = NLOGSPACE.

$\perp^{z}_{\sigma,h}$. The fact that $J_i\langle \mathsf{o}, \sigma_i, h_i\rangle J_{i+1}$ and (σ_i, h) is obliviously active, for each $i \geq 0$, follows by construction and the observation that, for each TGD $\sigma'_i \in \mathsf{enrichment}(\Sigma)$ for $i \geq 0$, $\mathsf{fr}(\sigma'_i)$ coincides with $\mathsf{var}(\mathsf{body}(\sigma_i))$. □

It is also crucial to observe that the class of sticky sets of TGDs is closed under enrichment. In other words:

Lemma 6 *For each set* $\Sigma \in \mathbb{S}$, $\mathsf{enrichment}(\Sigma) \in \mathbb{S}$.

Proof Consider a set $\Sigma \in \mathbb{S}$. The claim follows from the fact that, for each $\sigma \in \Sigma$ of the form $\phi(\bar{x}, \bar{y}) \rightarrow \exists \bar{z}\ \psi(\bar{x}, \bar{z})$, the corresponding TGD $\sigma' \in \mathsf{enrichment}(\Sigma)$ of the form $\phi(\bar{x}, \bar{y}) \rightarrow \exists \bar{z}\ \psi(\bar{x}, \bar{z})$, $\mathrm{Aux}_\sigma(\bar{x}, \bar{y})$ is such that the atom $\mathrm{Aux}_\sigma(\bar{x}, \bar{y})$ contains every body variable of σ, and no TGD in $\mathsf{enrichment}(\Sigma)$ has an atom with predicate Aux_σ in its body. Hence, no variable in $\mathsf{enrichment}(\Sigma)$ is marked. □

From Lemmas 5 and 6, we can conclude that $\mathsf{CT}^{\mathsf{o}}_{\forall}(\mathbb{S})$ can be reduced in logspace to $\mathsf{CT}^{\mathsf{so}}_{\forall}(\mathbb{S})$. Since, as shown above, $\mathsf{CT}^{\mathsf{o}}_{\forall}(\mathbb{S})$ is in PSPACE, and PSPACE is closed under logspace reductions, we immediately get that $\mathsf{CT}^{\mathsf{o}}_{\forall}(\mathbb{S})$ is in PSPACE, as needed. However, in the case of predicates of bounded arity, we cannot immediately inherit the NLOGSPACE upper bound since the reduction provided by Lemmas 5 and 6 introduces auxiliary predicates of the form Aux_σ, where σ is a TGD, of unbounded arity. Indeed, the arity of this auxiliary predicate is the number of variables occurring in the body of the TGD σ, which can be unbounded even if we use only bounded-arity predicates. Nevertheless, a predicate Aux_σ does not occur in the body of a TGD, which means that it cannot be part of a cycle in the underlying dependency graph. Therefore, we can consider a slightly modified version of Algorithm 1 that simply ignores the atoms that mention a predicate Aux_σ. By relying on this modified algorithm, we get that for a set Σ of TGDs over predicates of bounded arity, checking whether $\mathsf{Lin}(\mathsf{enrichment}(\Sigma))$ is not critically-weakly-acyclic is in NLOGSPACE, which implies that $\mathsf{CT}^{\mathsf{o}}_{\forall}(\mathbb{S})$ is in NLOGSPACE for predicates of bounded arity.

6.3 Lower Bounds

We conclude the proof of Theorem 2 by providing matching lower bounds. We show that $\mathsf{CT}^{\mathsf{o}}_{\forall}(\mathbb{S})$ and $\mathsf{CT}^{\mathsf{so}}_{\forall}(\mathbb{S})$ are PSPACE-hard, and NLOGSPACE-hard for predicates of bounded arity. Let us start with the general case of unbounded-arity predicates.

Predicates of Unbounded Arity Since, as discussed above, $\mathsf{CT}^{\mathsf{o}}_{\forall}(\mathbb{S})$ can be reduced in logspace to $\mathsf{CT}^{\mathsf{so}}_{\forall}(\mathbb{S})$, it suffices to show that $\mathsf{CT}^{\mathsf{o}}_{\forall}(\mathbb{S})$ is PSPACE-hard, or, equivalently, its complement is PSPACE-hard. We show that every problem in PSPACE can be reduced in logspace to the complement of $\mathsf{CT}^{\mathsf{o}}_{\forall}(\mathbb{S})$. Fix a problem Π in PSPACE, and let $M = (S, \Lambda, f, s_1)$ be the deterministic polynomial space Turing machine that solves Π, where $S = \{s_1, \ldots, s_q\}$ is the set of states of M, $\Lambda = \{0, 1, \sqcup\}$ is the tape alphabet of M with $\sqcup$ being the blank symbol, $f : S \times \Lambda \rightarrow (S \times \Lambda \times \{\leftarrow, -, \rightarrow\})$ is the transition function of M, and $s_1 \in S$ is the initial state. We assume, w.l.o.g., that M is well-behaved and never tries to read beyond its tape boundaries, always halts,

and uses exactly $n = m^k$ tape cells, where $k > 0$ and m is the size of the input string. We also assume that the machine accepts if it reaches a configuration with state s_2. For the purposes of the current proof, we represent a configuration of M as a string

$$s, t_1, c_1, t_2, c_2, \ldots, t_n, c_n,$$

where $s \in S$, $(t_i, c_i) \in \Lambda \times \{\uparrow, \flat\}$, for each $i \in [n]$, and there is exactly one $i \in [n]$ such that $c_i = \uparrow$. Such a string encodes the configuration where the state is s, the i-th cell of the machine contains the symbol t_i, and, assuming that $c_i = \uparrow$, the cursor is on the i-th cell of the machine. For example, the initial configuration of M on input $I = a_1, \ldots, a_m$ is

$$s_1, a_1, \uparrow, a_2, \flat, \ldots, a_m, \flat, \underbrace{\sqcup, \flat, \ldots, \sqcup, \flat}_{n-m}.$$

Our goal is to construct a set $\Sigma \in \mathbb{S}$ such that M accepts on input $I = a_1, \ldots, a_m$ iff $\Sigma \notin \mathbb{CT}^{\mathsf{o}}_{\forall}$, or, equivalently, there exists an infinite o-chase derivation of $\mathsf{cr}(\Sigma)$ w.r.t. Σ. The high-level idea is, starting from an atom of the form $\mathrm{Start}(\natural, \bot, \natural)$, where $\bot$ is a null, to apply a TGD σ_{start} and generate the initial configuration of M on input I, which will be stored in a predicate Config. Then, each application of a TGD will mimic a transition rule of f and generate a valid configuration of M. Once an accepting configuration is reached, then an atom of the form $\mathrm{Start}(\bot, \bot', \natural)$, where $\bot'$ is a null different than $\bot$, will be generated, which will trigger again the TGD σ_{start}. This will give rise to an infinite o-chase derivation of $\mathsf{cr}(\Sigma)$ w.r.t. Σ. To achieve this, however, via a sticky set of TGDs, we need a proper encoding of a configuration of M as a Config-atom generated during the execution of the chase. We proceed to explain this encoding, which will then allow us to define our sticky set of TGDs.

The key idea is to encode a state of M, the symbols of Λ, and the symbols $\uparrow, \flat$, as tuples consisting of a null $\bot \in \mathbf{N}$ and a single occurrence of the constant $\natural$, while the position at which $\natural$ occurs in this tuple identifies the state or symbol in question. In particular, a state $s_i \in S$, for $i \in [q]$, will be represented in the chase as a tuple

$$\underbrace{\bot, \ldots, \bot}_{i-1}, \natural, \underbrace{\bot, \ldots, \bot}_{|S|-i}.$$

The tape alphabet 0, 1, $\sqcup$ will be represented as

$$\natural, \bot, \bot \qquad \bot, \natural, \bot \qquad \bot, \bot, \natural,$$

respectively. Finally, the symbols $\uparrow, \flat$ will be represented as

$$\natural, \bot \qquad \bot, \natural,$$

respectively. For example, assuming that $I = 1, 1, \ldots, 1, 0$, that is, $a_i = 1$, for each $i \in [m-1]$, and $a_m = 0$, the initial configuration of M on input I will be represented in the chase via an atom of the form $\mathrm{Config}(\bar{t}_1, \bar{t}_2, \bar{t}_3)$, where

$$\bar{t}_1 = \overbrace{\natural, \underbrace{\bot, \ldots, \bot}_{|S|-1}}^{s_1}$$

$$\bar{t}_2 \;=\; \overbrace{\bot, \natural, \bot}^{1}, \overbrace{\natural, \bot}^{\uparrow}, \underbrace{\overbrace{\bot, \natural, \bot}^{1}, \overbrace{\bot, \natural}^{\flat}, \ldots, \overbrace{\bot, \natural, \bot}^{1}, \overbrace{\bot, \natural}^{\flat}}_{m-2}, \overbrace{\natural, \bot, \bot}^{0}, \overbrace{\bot, \natural}^{\flat},$$

$$\bar{t}_3 \;=\; \underbrace{\overbrace{\bot, \bot, \natural}^{\sqcup}, \overbrace{\bot, \natural}^{\flat}, \ldots, \overbrace{\bot, \bot, \natural}^{\sqcup}, \overbrace{\bot, \natural}^{\flat}}_{n-m}$$

We are now ready to devise the desired set Σ of TGDs. For notional convenience, we are going to use the following abbreviations for tuples of variables:

$$\text{for each } i \in [q],\ \boldsymbol{s_i} \;=\; \underbrace{y, \ldots, y}_{i-1}, z, \underbrace{y, \ldots, y}_{|S|-i}$$

$$\mathbf{0} \;=\; z, y, y \qquad \mathbf{1} \;=\; y, z, y \qquad \sqcup \;=\; y, y, z \qquad \uparrow \;=\; z, y \qquad \flat \;=\; y, z$$

Observe that the above abbreviations follow exactly the same pattern as the encodings described above. In fact, whenever we need to refer to the state s_i in a TGD, we are going to use the abbreviation $\boldsymbol{s_i}$, which means that the variable y will be mapped to a null, and the variable z to the constant $\natural$. Analogously, if we need to refer to a tape symbol or to $\uparrow, \flat$, we are going to use the abbreviations $\mathbf{0}$, $\mathbf{1}$, $\sqcup$, $\uparrow$ and $\flat$, respectively. The set Σ consists of the following TGDs:

$$\text{Start}(x, y, z) \rightarrow \text{Config}(\boldsymbol{s_1}, \boldsymbol{a_1}, \uparrow, \boldsymbol{a_2}, \flat, \ldots, \boldsymbol{a_m}, \flat, \underbrace{\sqcup, \flat, \ldots, \sqcup, \flat}_{n-m}, y, z)$$

which generates the initial configuration of M on input $I = a_1, \ldots, a_m$. The rest of the TGDs are responsible for simulating the transition function of M.

Before defining those TGDs, let us introduce some additional abbreviations. Let

$$\text{for each } i \in [n],\ \boldsymbol{x_i} \;=\; x_i^0, x_i^1, x_i^{\sqcup} \quad \text{and} \quad \boldsymbol{x_i^c} \;=\; x_i^{\uparrow}, x_i^{\flat}$$

which will be used as placeholders for the content of the i-th cell, and the position of the cursor. In fact, the above abbreviations will simply help us to copy the unchanged cells from a configuration to its subsequent one. In addition, for $t \in S \cup \Lambda \cup \{\uparrow, \flat\}$, we write $\hat{\boldsymbol{t}}$ for the tuple of variables obtained from the tuple $\boldsymbol{t}$ after replacing each occurrence of the variable y at some position i with the new variable y_t^i, while the variable z remains unchanged. More precisely:

$$\text{for each } i \in [q],\ \hat{\boldsymbol{s_i}} \;=\; y_{s_i}^1, \ldots, y_{s_i}^{i-1}, z, y_{s_i}^{i+1}, \ldots, y_{s_i}^q$$

$$\hat{\mathbf{0}} = z, y_0^1, y_0^2 \qquad \hat{\mathbf{1}} = y_1^1, z, y_1^2 \qquad \hat{\sqcup} = y_{\sqcup}^1, y_{\sqcup}^2, z \qquad \hat{\uparrow} = z, y_{\uparrow}^1 \qquad \hat{\flat} = y_{\flat}^1, z$$

The above abbreviations will allow us to break some unnecessary joins over the variable y in the bodies of the TGDs, which in turn will ensure that the obtained set of TGDs is sticky. We proceed to introduce the rest of the TGDs.

We first introduce several TGDs that simulate the transition rules of f where the cursor moves to the left. In particular, for each transition rule $f(s_i, a) = (s_j, b, \leftarrow)$, and for each $\ell \in [n]$, we have the TGD

$$\text{Config}(\hat{\boldsymbol{s_i}}, \boldsymbol{x_1}, \boldsymbol{x_1^c}, \ldots, \boldsymbol{x_{\ell-1}}, \boldsymbol{x_{\ell-1}^c}, \hat{\boldsymbol{a}}, \hat{\uparrow}, \boldsymbol{x_{\ell+1}}, \boldsymbol{x_{\ell+1}^c}, \ldots, \boldsymbol{x_n}, \boldsymbol{x_n^c}, y, z) \;\rightarrow$$
$$\text{Config}(\boldsymbol{s_j}, \boldsymbol{x_1}, \boldsymbol{x_1^c}, \ldots, \boldsymbol{x_{\ell-1}}, \uparrow, \boldsymbol{b}, \flat, \boldsymbol{x_{\ell+1}}, \boldsymbol{x_{\ell+1}^c}, \ldots, \boldsymbol{x_n}, \boldsymbol{x_n^c}, y, z).$$

Analogously, we have TGDs that simulate the transition rules of f where the cursor moves to the right. In particular, for each transition rule $f(s_i, a) = (s_j, b, \rightarrow)$, and for each $\ell \in [n]$, we have the TGD

$$\mathrm{Config}(\hat{\boldsymbol{s_i}}, \boldsymbol{x_1}, \boldsymbol{x_1^c}, \ldots, \boldsymbol{x_{\ell-1}}, \boldsymbol{x_{\ell-1}^c}, \hat{\boldsymbol{a}}, \hat{\uparrow}, \boldsymbol{x_{\ell+1}}, \boldsymbol{x_{\ell+1}^c}, \ldots, \boldsymbol{x_n}, \boldsymbol{x_n^c}, y, z) \rightarrow$$
$$\mathrm{Config}(\boldsymbol{s_j}, \boldsymbol{x_1}, \boldsymbol{x_1^c}, \ldots, \boldsymbol{x_{\ell-1}}, \boldsymbol{x_{\ell+1}^c}, \boldsymbol{b}, \mathfrak{b}, \boldsymbol{x_{\ell+1}}, \uparrow, \boldsymbol{x_{\ell+2}}, \boldsymbol{x_{\ell+2}^c}, \ldots, \boldsymbol{x_n}, \boldsymbol{x_n^c}, y, z).$$

We complete the simulation of the transition function by adding TGDs that simulate the transition rules of f where the cursor stays at the same position. In particular, for each transition rule $f(s_i, a) = (s_j, b, -)$, and for each $\ell \in [n]$, we have the TGD

$$\mathrm{Config}(\hat{\boldsymbol{s_i}}, \boldsymbol{x_1}, \boldsymbol{x_1^c}, \ldots, \boldsymbol{x_{\ell-1}}, \boldsymbol{x_{\ell-1}^c}, \hat{\boldsymbol{a}}, \hat{\uparrow}, \boldsymbol{x_{\ell+1}}, \boldsymbol{x_{\ell+1}^c}, \ldots, \boldsymbol{x_n}, \boldsymbol{x_n^c}, y, z) \rightarrow$$
$$\mathrm{Config}(\boldsymbol{s_j}, \boldsymbol{x_1}, \boldsymbol{x_1^c}, \ldots, \boldsymbol{x_{\ell-1}}, \boldsymbol{x_{\ell-1}^c}, \boldsymbol{b}, \uparrow, \boldsymbol{x_{\ell+1}}, \boldsymbol{x_{\ell+1}^c}, \ldots, \boldsymbol{x_n}, \boldsymbol{x_n^c}, y, z).$$

Finally, we introduce a TGD that checks whether an accepting configuration has been reached; recall that, by assumption, the machine accepts if it reaches a configuration with state s_2:

$$\mathrm{Config}(\hat{\boldsymbol{s_2}}, \boldsymbol{x_1}, \boldsymbol{x_1^c}, \ldots, \boldsymbol{x_n}, \boldsymbol{x_n^c}, y, z) \rightarrow \exists w \mathrm{Start}(y, w, z).$$

Observe that, if an accepting configuration is reached, then a new atom of the form $\mathrm{Start}(\_, \bot, \natural)$ is obtained, where $\bot$ is a new null value, which will trigger again the first TGD that generates the initial configuration. This completes the definition of Σ.

It is now easy to see that M accepts on input I iff there exists an infinite o-chase derivation of $\mathsf{cr}(\Sigma)$ w.r.t. Σ, or, equivalently, $\Sigma \notin \mathbb{CT}^{\mathsf{o}}_{\forall}$. Let us remark that starting from $\mathsf{cr}(\Sigma)$, the first TGD that will be applied is the one with the Start predicate in the head, and an atom α of the form $\mathrm{Start}(\natural, \bot, \natural)$, where $\bot$ is a null, will be generated. Then, α will trigger the TGD that generates the initial configuration of M on input I, which will then guarantee a faithful simulation of the computation of M on I. Moreover, it is easy to verify that $\Sigma \in \mathbb{S}$. The key reason why this holds is because the only variable that occurs more than once in the body of a TGD is z, which is always propagated to the head at the same position, which is actually the last one.

Predicates of Bounded Arity. By Lemmas 5 and 6, and the definition of enrichment, for every set $\Sigma \in \mathbb{S}$, it holds that:

- $\Sigma \in \mathbb{CT}^{\mathsf{o}}_{\forall}$ iff $\mathsf{enrichment}(\Sigma) \in \mathbb{CT}^{\mathsf{so}}_{\forall}$.
- $\mathsf{enrichment}(\Sigma) \in \mathbb{S}$.
- $\mathsf{ar}(\mathsf{enrichment}(\Sigma)) = \max\{\mathsf{ar}(\Sigma), k\}$, where $k \geq 0$ is the maximum number of variables occurring in the body of a TGD of Σ.

Therefore, to show that $\mathsf{CT}^{\mathsf{o}}_{\forall}(\mathbb{S})$ and $\mathsf{CT}^{\mathsf{so}}_{\forall}(\mathbb{S})$ are NLogSpace-hard for predicates of bounded arity, it suffices to concentrate on $\mathbb{CT}^{\mathsf{o}}_{\forall}$ and provide a hardness result even for TGDs that can have at most k body variables for some fixed integer $k \geq 0$. This can be easily shown via a logspace reduction from graph reachability. Consider a directed graph $G = (N, E)$, and two nodes $s, t \in N$. The problem of deciding whether in G the node t is reachable from s is NLogSpace-hard. We are going to construct a sticky set Σ of TGDs that uses only unary predicates, and each TGD has only one body variable, such that t is reachable from s in G iff $\Sigma \notin \mathbb{CT}^{\mathsf{o}}_{\forall}$. This in turn

implies that $\mathsf{CT}^{\mathsf{o}}_{\forall}(\mathbb{S})$ is NLogSpace-hard since coNLogSpace = NLogSpace. The set Σ consists of the following TGDs:

$$\begin{aligned} \mathrm{Loop}(x) &\rightarrow P_s(x) \\ P_v(x) &\rightarrow P_u(x), \text{ for each } (v,u) \in E \\ P_t(x) &\rightarrow \exists z\, \mathrm{Loop}(z). \end{aligned}$$

It is easy to see that indeed t is reachable from s iff $\Sigma \notin \mathbb{CT}^{\mathsf{o}}_{\forall}$, and the claim follows.

7 Discussion and Future Work

We have studied all-instances (semi-)oblivious chase termination for sticky sets of TGDs, and provide precise complexity results. It turned out that our results and techniques allow us to obtain further complexity results concerning the chase procedure.

Further Results Interestingly, our main result has already found applications in the context of (semi-)oblivious *chase boundedness*, which has been recently studied in [7]. Roughly, chase boundedness guarantees, not only the termination of the chase procedure, but also the existence of a uniform bound on the depth of the chase. As shown in [7], in the case of sticky sets of TGDs, (semi-)oblivious chase termination and (semi-)oblivious chase boundedness coincide. Therefore, from the main result of our work, we immediately get the complexity of deciding (semi-)oblivious chase boundedness in the case of sticky sets of TGDs.

Moreover, the techniques that have been developed in this work can also be applied to *shy* sets of TGDs, another well-known TGD-based formalism [25, 26]. Shy sets of TGDs are not powerful enough to join over null values that have been invented during the execution of the chase. This central property of shy sets of TGDs allows us to reuse the machinery developed for stickiness, and show that the (semi-)oblivious chase termination problem for shy sets of TGDs is PSpace-complete in general, and NLogSpace-complete for predicates of bounded arity.

Future Work Although all-instances (semi-)oblivious chase termination for sticky sets of TGDs is well-understood, there are still interesting and highly non-trivial open questions that we are planning to address in the near future:

1. Whenever the (semi-)oblivious chase terminates under a sticky set of TGDs, what is the exact size of the final result?
2. Even if the (semi-)oblivious chase does not terminate, it might be the case that a finite universal model exists [15]. Can we decide, in the case of sticky sets of TGDs, whether a finite universal model exists? And what about its size?

Acknowledgements This work has been funded by the EPSRC grant EP/S003800/1 "EQUID", and the EPSRC programme grant EP/M025268/ "VADA".

References

1. Abiteboul, S., Hull, R., Vianu, V.: Foundations of Databases. Addison-Wesley, Reading, MA (1995)
2. Aho, A.V., Sagiv, Y., Ullman, J.D.: Efficient optimization of a class of relational expressions. ACM Trans. Database Syst. **4**(4), 435–454 (1979)
3. Baget, J., Mugnier, M., Rudolph, S., Thomazo, M.: Walking the Complexity Lines for Generalized Guarded Existential Rules. In: IJCAI, pp. 712–717 (2011)
4. Baget, J.F., Leclère, M., Mugnier, M.L., Salvat, E.: On rules with existential variables: walking the decidability line. Artif. Intell. **175**(9-10), 1620–1654 (2011)
5. Beeri, C., Vardi, M.Y.: A proof procedure for data dependencies. J. ACM **31**(4), 718–741 (1984)
6. Benedikt, M., Konstantinidis, G., Mecca, G., Motik, B., Papotti, P., Santoro, D., Tsamoura, E.: Benchmarking the Chase. In: PODS, pp. 37–52 (2017)
7. Bourhis, P., Leclère, M., Mugnier, M., Tison, S., Ulliana, F., Gallois, L.: Oblivious and Semi-Oblivious Boundedness for Existential Rules. In: IJCAI, pp. 1581–1587 (2019)
8. Calautti, M., Gottlob, G., Pieris, A.: Chase Termination for Guarded Existential Rules. In: PODS, pp. 91–103 (2015)
9. Calautti, M., Greco, S., Molinaro, C., Trubitsyna, I.: Exploiting equality generating dependencies in checking chase termination. PVLDB **9**(5), 396–407 (2016)
10. Calautti, M., Pieris, A.: Oblivious Chase Termination: The Sticky Case. In: ICDT, pp. 17:1–17:18 (2019)
11. Calì, A., Gottlob, G., Kifer, M.: Taming the infinite chase: query answering under expressive relational constraints. J. Artif. Intell. Res. **48**, 115–174 (2013)
12. Calì, A., Gottlob, G., Lukasiewicz, T.: A general Datalog-based framework for tractable query answering over ontologies. J. Web Sem. **14**, 57–83 (2012)
13. Calì, A., Gottlob, G., Pieris, A.: Towards more expressive ontology languages: the query answering problem. Artif. Intell. **193**, 87–128 (2012)
14. Cuenca Grau, B., Horrocks, I., Krötzsch, M., Kupke, C., Magka, D., Motik, B., Wang, Z.: Acyclicity notions for existential rules and their application to query answering in ontologies. J. Artif. Intell. Res. **47**, 741–808 (2013)
15. Deutsch, A., Nash, A., Remmel, J.B.: The Chase Revisisted. In: PODS, pp. 149–158 (2008)
16. Deutsch, A., Tannen, V.: Reformulation of XML Queries and Constraints. In: ICDT, pp. 225–241 (2003)
17. Fagin, R., Kolaitis, P.G., Miller, R.J., Popa, L.: Data exchange: semantics and query answering. Theor. Comput. Sci. **336**(1), 89–124 (2005)
18. Gogacz, T., Marcinkowski, J.: All-Instances Termination of Chase is Undecidable. In: ICALP, pp. 293–304 (2014)
19. Gogacz, T., Marcinkowski, J., Pieris, A.: All-Instances Restricted Chase Termination. In: PODS (2020). To appear
20. Grahne, G., Onet, A.: Anatomy of the chase. Fundam. Inform. **157**(3), 221–270 (2018)
21. Greco, S., Spezzano, F., Trubitsyna, I.: Stratification criteria and rewriting techniques for checking chase termination. PVLDB **4**(11), 1158–1168 (2011)
22. Krötzsch, M., Marx, M., Rudolph, S.: The Power of the Terminating Chase (Invited Talk). In: ICDT, pp. 3:1–3:17 (2019)
23. Krötzsch, M., Rudolph, S.: Extending Decidable Existential Rules by Joining Acyclicity and Guardedness. In: IJCAI, pp. 963–968 (2011)

24. Leclère, M., Mugnier, M., Thomazo, M., Ulliana, F.: A Single Approach to Decide Chase Termination on Linear Existential Rules. In: ICDT, pp. 18:1–18:19 (2019)
25. Leone, N., Manna, M., Terracina, G., Veltri, P.: Efficiently Computable Datalog∃ Programs. In: KR (2012)
26. Leone, N., Manna, M., Terracina, G., Veltri, P.: Fast query answering over existential rules. ACM Trans. Comput. Log. **20**(2), 12:1–12:48 (2019)
27. Maier, D., Mendelzon, A.O., Sagiv, Y.: Testing implications of data dependencies. ACM Trans Database Syst. **4**(4), 455–469 (1979)
28. Marnette, B.: Generalized Schema-Mappings: from Termination to Tractability. In: PODS, pp. 13–22 (2009)
29. Meier, M., Schmidt, M., Lausen, G.: On chase termination beyond stratification. PVLDB **2**(1), 970–981 (2009)
30. Nenov, Y., Piro, R., Motik, B., Horrocks, I., Wu, Z., Banerjee, J.: Rdfox: a Highly-Scalable RDF Store. In: ISWC, pp. 3–20 (2015)
31. Papadimitriou, C.H.: Computational complexity. Addison-Wesley, Reading, MA (1994)
32. Rudolph, S., Krötzsch, M., Hitzler, P.: All Elephants are Bigger than All Mice. In: DL (2008)
33. Urbani, J., Krötzsch, M., Jacobs, C.J.H., Dragoste, I., Carral, D.: Efficient Model Construction for Horn Logic with Vlog - System Description. In: IJCAR, pp. 680–688 (2018)